NTNU

Norwegian University of
Science and Technology

Coordination of Teams in Large-scale Agile Software Development Projects

An exploratory case study

Kamilla Fredriksen

Master of Science in Computer Science
Submission date: May 2018
Supervisor: Torgeir Dingsøyr, IDI

Norwegian University of Science and Technology
Department of Computer Science

Summary

Agile software development methods value individuals and interactions, working software and customer collaboration. These methods have been believed to suit small, co-located projects best. Success in small projects has lead to application of agile methods in larger projects as well. However, since these methods initially were developed for small projects, fundamental assumptions of the agile software development are challenged when used in large-scaled projects.

One main issue in large-scale projects raised in several scientific publications, is the challenge of coordination of teams. The overall aim of this thesis was to study how large organisations can manage inter-team coordination of large-scale agile software development projects in practice.

This study was conducted as a literature review focused on providing background information and historical interpretation of the subject, and a single case study; that was considered as one of the largest agile software development projects taking place in Norway at this time. The investigated project was expected to have duration of about two years. It consisted of four co-located development teams and involved a total of 132 people. The case study included an analysis of 12 semi-structured interviews and documents, with a total of 188 transcribed pages. Data provided through the analysis was qualitatively categorised according to a predefined model of coordination based on prevailing theory by Van de Ven.

Findings showed that large-scale agile software development projects can handle inter-team coordination through the use of widespread informal communication, hierarchical meeting structure, a central unit responsible for coordination, structure and standardisation of tasks and work from the beginning, specialised and co-located teams, and maintaining the agility and flexibility of the project.

Sammendrag

Smidige programvareutviklingsmetoder verdsetter enkeltpersoner og samspill, fungerende programvare og kundesamarbeid. Disse metodene er antatt å passe små, samlokaliserte prosjekter best. Suksess i små prosjekter har ført til anvendelse av smidige metoder i større prosjekter. Siden disse metodene i utgangspunktet ble utviklet for små prosjekter, utfordres grunnleggende forutsetninger for den smidige programvareutviklingen når den brukes i større projecter.

Et hovedproblem i større prosjekter som er påvist i flere vitenskapelige publikasjoner, er utfordringen med koordinering av team. Det overordnede målet med denne oppgaven var å studere hvordan store organisasjoner kan styre koordinering på tvers av team i store, smidige programvareutviklingsprosjekter i praksis.

Denne studien ble utført som en litteratur gjennomgang fokusert på å gi bakgrunnsinformasjon og historisk tolkning av emnet, og et enkelt case-studie; som ble ansett som en av de største, smidige programvareutviklingsprosjektene i Norge på dette tidspunktet. Det undersøkte prosjektet ble forventet å ha en varighet på om lag to år. Det besto av fire samlokaliserte utviklingsgrupper og involverte totalt 132 personer. Case-studien omfattet en analyse av 12 semistrukturerte intervjuer og dokumenter, med totalt 188 transkriberte sider. Data gitt gjennom analysen ble kvalitativt kategorisert i følge en forhåndsdefinert koordinasjonsmodell basert på rådende teori til Van de Ven.

Resultatene viste at store, smidige programvareutviklingsprosjekter kan håndtere koordinering på tvers av team ved bruk av utbredt uformell kommunikasjon, en hierarkisk møte struktur, en sentral enhet ansvarlig for koordinering, struktur og standardisering av oppgaver og arbeid fra begynnelsen, spesialiserte og samlokaliserte team, og opprettholde fleksibiliteten i prosjektet.

Acknowledgment

This master thesis is written during Spring 2018 as part of the Master program in Computer Science at the Department of Computer Science (IDI) at the Norwegian University of Science and Technology (NTNU). My specialisation within computer science has been software development. This specialisation has given me insight to methodologies used when developing software products, which has cultured my interest for methodologies used in software development. In addition I have communicated with the businesses working with software development throughout my education, where issues related to large-scale agile project has been a subject. Thereby I chose to research the field of agile software development in large-scale projects. Because I found that inter-team coordination was stated in several scientific papers to be an issue in these projects, I decided to investigate this subject further. However, the chosen subject for analysis was found to be sparingly researched, which made the analysis more challenging.

I will therefore like to thank Torgeir Dingsøy for given me helpful advises, and being a motivator throughout my work with this master thesis, but still given me time to work independently. His knowledge and experience with the field of study has been found very inspiring, helpful and appreciated.

K.K.F.

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Introduction

1.1 Problem description, background and aim

The society of today is constantly changing. The technology is developing exponentially, becoming increasingly complex. Thus, this rapid change in society and technology has led to changes in the evolution of software development processes as well.

An approach to software development that has gained growing interest is the agile methodology. This methodology can be defined with reference to a set of principles for software development elaborated in the publication of the Agile Manifesto for agile software development (Beck et al. (2001)); a declaration of four values and twelve principles. It doesn't provide concrete steps, and manufacturers usually seek more specific methods within the Agile movement; such as Crystal Clear, Extreme Programming, Feature Driven Development, Dynamic Systems Development Method (DSDM), Scrum, and others. The Agile Manifesto was developed for the modern society, primarily in order to adapt rapidly and flexibly to changes. The agile methodology is referred to as a methodology that values individuals and interactions, working software and customer collaboration, early delivery, and continuous improvement (Larman (2004)).

Since the agile paradigm was released in 2001, it has become the most widely used methodology for software development (VersionOne (2017)). The methodology has also become an attractive field of research in order to find improvements in software processes.

Although the agile methodology was originally meant as a solution for small-scale projects, it has during the past few years been transferred to even more large-scaled software development projects. The research on the use of this methodology in large project is however scarce (Dingsøyr

and Moe (2014), Scheerer and Kude (2014)). Studies have indicated that there has been an improvement in the quality of the software in small to medium sized projects. The improvements could be explained as a result of increased use of agile methodologies in organisations.

However, the increased use of agile methodologies does not come without challenges, especially when used in a large-scaled software development context. Despite this, the number of large organisations using agile software development has increased. A survey conducted by the VersionOne showed that from 2016 to 2017 the number of large organisations that are using agile software development has increased from 24% to 26% (VersionOne (2017)). One main issue related to large-scale agile software development, which has been raised in several scientific publications, is the challenge of coordination between teams (Dingsøyr and Moe (2014), Dingsøyr et al. (2017), Bick et al. (2008), Bick et al. (2016), Paasivaara et al. (2012).

The aim for this study is to explore how teams are being coordinated in large-scale agile software development. Research literature has delivered evidence for smaller agile projects to be coordinated mainly by self-management (Rising and Janoff (2000)). However, this may seem impracticable in a larger organisation. As the teams tend to get larger and more complex, the role of self-management has been shown to reduce the ability to coordinate across teams effectively (Ingvaldsen and Rolfsen (2012)). Also the research on this field, and the theoretical grounding from empirical research has been scarce (Scheerer and Kude (2014), Dingsøyr and Moe (2014)). Therefore, this study aim to increase insight and contribute to the agile at scale research, conducting a single-case study. A real life case will be analysed in terms of inter-team coordination in large-scale agile software development projects.

1.2 Personal motivation

Autumn 2017 I wrote a project thesis, where I conducted a literature review on how coordination has been done in agile software development until today. Through the thesis I identified several mechanisms that the employees seemed to be satisfied with. These mechanisms mainly concerned coordination in terms of optimising communication and structure of the project. However, in several projects found, the employees were not satisfied with the coordination, and when these projects had been followed up later on, they still did not know how to improve their way of coordinating these projects. Writing on this project thesis made me curious of how coordination issues of large-scale agile software development projects are solved today. This is also what I attempt to answer through this master thesis by following a large-scale agile software development

project, analysing interviews and making observations.

1.3 Research questions

With the aim to explore this field, inter-team coordination in large-scale agile software development projects, this brings us to formally describing what question this research intends to answer. There is a gap of knowledge and a lack of research within large-scale agile software development projects, and coordination is an important factor for success within these projects. Therefore, this thesis will try to contribute with extended knowledge regarding coordination of large-scale agile software development projects. The main research question for this research will attempt to be answered is therefore:

> ***"How can large organisations manage inter-team coordination of large-scale agile software development projects in practice?"***

With this main question this study will look deeper into the inter-team coordination through analysis of a case. The investigated case is considered of being large-scale, meaning that it has more than one team, but less then than ten. In this case they also used agile mechanisms, where each team used the Scrum methodology. In order to answer the research question and give further insight into inter-team coordination of agile software development projects, the following sub-questions are raised:

> **Sub-question 1:** What does the prevailing literature tells us about inter-team coordination in large-scale agile software development projects?
> **Sub-question 2:** Which coordination mechanisms do the investigated project use in order to coordinate?

Through the answers to the research questions above, the thesis should contribute to explore mechanisms and work out strategies for practice that support and optimise inter-team coordination in large-scale agile software development projects. The answers could also be useful to serve as indicators on which challenges one may face when managing inter-team coordination in such large scale projects.

1.4 Research scope

This paper is a master thesis conducted for the Department of Information and Electrical Engineering at the Norwegian University of Science and Technology written by Kamilla Kristine Fredriksen. The research was conducted during the spring of 2018 and is worth 30 points in the European Credit Transfer and Accumulation System (ECTS), which estimates to 750-900 working hours [1]. The time used includes planning, research, documentation and the writing related to this research.

1.5 Intended audience

The audience that has been focused on when writing this thesis are mainly of three types:

- **Researchers** can find this research interesting, since there is a limited amount of case studies conducted in this research area.
- **Practitioners** within the field of agile software development in large-scale projects can find this research interesting, as it can help them improve strategies, but also help them to avoid potential pitfalls.
- **Students** who research the field of agile software development in large-scale projects, may find the research interesting due to lack of studies conducted on large-scaled projects and the use of agile software development. In addition, this research illustrates how work is done in real-life practice and not only in theory, which is often found in the textbooks.

1.6 Structure of the thesis

In this thesis I will first introduce basic theory regarding software development methodology in Chapter 2, then I will present more deeply the term coordination, coordination of small projects and large projects according to some leading theories in Chapter 3. Thereafter, you will be introduced to the method for this thesis in order to gain its results in Chapter 4. Applying this method leads to the results which are reported in Chapter 5. These results will then be discussed, and also the research questions, model used and limitations will be evaluated in Chapter 6. Finally,

[1] http://www.mastersportal.eu/articles/388/all-you-need-to-know-about-the-european-credit-system-ects.html, Accessed: 30.01.18

you will get to the concluding remarks, which directly answer the research question in Chapter 7.

Part II

Theory

Chapter 2

Software development methodologies

This section aims to introduce the most prominent methodologies used when developing software. I believe it is important to get an understanding of both plan-driven and agile methodologies even though this research focuses on agile software development and mostly the Scrum. The reason for this is that when describing large projects, it is often reported that mechanisms from both methodologies are used.

2.1 The role of methodologies and frameworks

Before diving into different software methodologies it is well worth to clarify what a software methodology is. Sommerville (2010),p. 59,defines a *software development process* as:

"A software process is a set of related activities that leads to the production of a software product"

This definition does not use the term methodology, but it states that a software development process is mainly a product of systematic activities that leads to a product. An example of this types of activities is Royces (1987) first model of the software development process, the Waterfall methodology, which suggests that two activities, namely Analysis and Coding, is enough to be a software development process. The term *methodology* is, further, defined in the Cambridge Dictionaries[1] as *"a system of ways of doing, teaching, or studying something".*

A *software development methodology* is then, by combining these two definitions, a systematic way

[1]https://dictionary.cambridge.org/dictionary/english/methodology Accessed: 10.01.18

of doing a software development process. The term *"method"* is also used when talking about a methodology, and some even call it a *framework* with assumed emphasis on structural elements. This is despite the meaning of framework and methodology is basically quite different.

The definition of a *framework* is also by Cambridge Dictionary[2] defined as *"a supporting structure around which something can be built"*. Meaning that a methodology is more prescriptive than a framework, as it shall describe the different steps that shall be taken. A *framework* on the other hand, only gives the frame, and hence the term and the use of it, is quite flexible in how it decides to develop within the frame that is unspecified or variable content.

2.2 Plan-driven software development

One of the first methodologies used in software development projects was the *traditional software development* that is also known as *plan-driven software development*. This methodology is characterised by how it consists of separate sequential stages representing the software development process. In addition, the methodology is known for planning before development, and its hierarchy and central decision-making. In this methodology, the product from one stage is used as a basis in the next, and there is no room for iterations across stages, as iterations are only allowed within one stage. One of the best known plan-driven software development models today, is the Waterfall model (Sommerville (2010)).

2.2.1 Waterfall

One researcher that explored the plan-driven development process, was Royce (1987). He released his research regarding software development processes, where he introduced the Waterfall methodology. The model is based on three principles: *lots of documentation, little customer involvement,* and *sequential structure of project realisation*. Further, this methodology has the characteristics of a plan-driven software development process, where each process stage is thoroughly planned, and follows linearly as a flow, shown in Figure 2.1. In this process decisions are done at the top hierarchical level. Further, one stage has to be finished before proceeding to the next. The input to a stage is the product from the former stage, and overlapping stages or going backwards are not allowed. It was from this structure that the methodology was given its name a few years later.

[2]https://dictionary.cambridge.org/dictionary/english/framework Accessed: 10.01.18

Royce's model is build on his assumption that it is not possible to do any implementation before the developer has insight into the problems that needs to be solved. This assumption gives two steps that all software development projects have to go through, namely Analysis and Coding. Even though this method sounds good in theory, Royce admitted that it still might fail in practice. He also suggested that such a simple model can only be used by small projects that are to be completed in a few days.

Later in his study, for larger projects, Royce suggested a model that includes five steps and allows for returning to previous steps. These steps are *Requirements Analysis, System and Software Design, Implementation and Unit Testing, Integration and System Testing,* and *Operations and Maintenance.* This is the model that is used in the Waterfall methodology nowadays. These five steps are illustrated in Figure 2.1 and listed and described in Table 2.1 below.

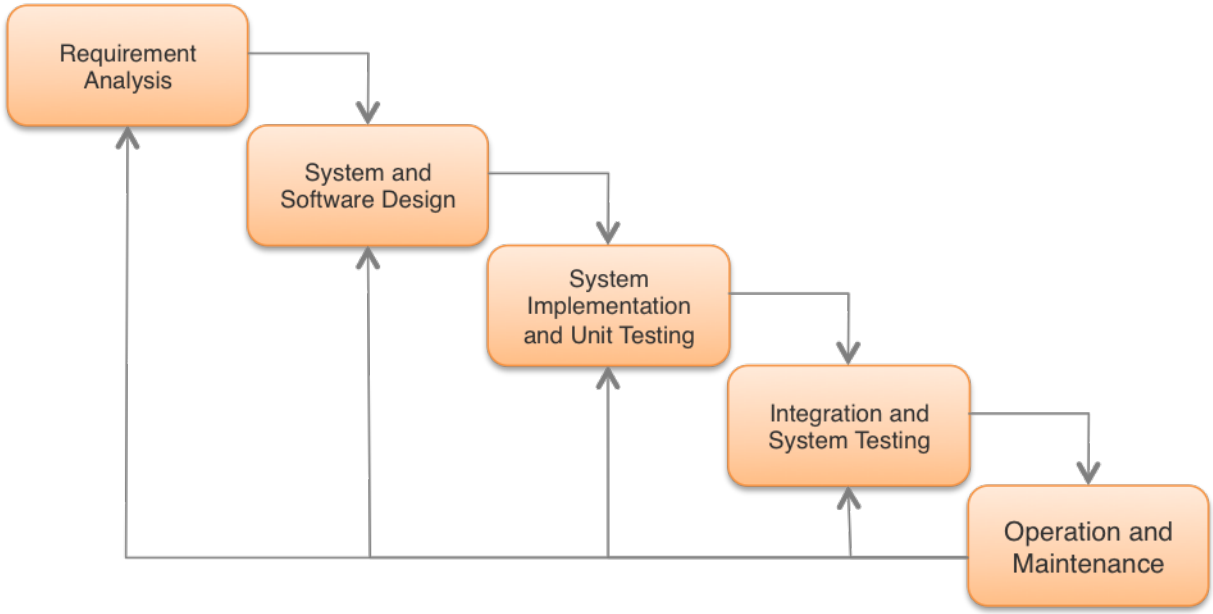

Figure 2.1: The Waterfall model (Adapted from Royce (1987))

Roles of Waterfall

The waterfall model calls for a hierarchical organisational structure, with decisions being made top-down. The model uses specialised teams and is considered quite large, with about 15 people or more. These teams do not tend to change throughout the project, but are rather static, as each of the team members has its defined area of responsibility. However, the roles in the teams tend to differ depending on which stage the project is operating in (Royce (1987)). For instance, the "requirement analysis"-phase requires different team roles than the "integration and

Table 2.1: The five steps of the Waterfall Model (Royce (1987))

Step	Description
Requirement Analysis	Collection of all types of requirements including functional, non-functional requirements and quality attribute requirements and documented in the requirements specification.
System and Software Design	A system specification is formed based on the requirements from the first face. Which helps for specifying system requirements and hardware.
Implementation and Unit Testing	The program is first developed in small pieces called units with the help of the system design. This step also includes unit testing.
Integration and System Testing	Units from the previous step are integrated to form a system and the system as a whole is tested.
Operation and Maintenance	Fixing issues that comes up after the system has been deployed.

system testing"-phase. The roles that are noticed in the Waterfall model are the following (Royce (1987)):

- *Business Analyst*: Responsible for constructing the requirements for the software being made together with the costumer and the project manager
- *Architect*: Responsible for the system and software design based on the requirements given from the business analysts.
- *Developer*: Responsible for the entire implementation of the system as well as unit testing, by following the system and software design given by the architects
- *Tester*: Responsible for conducting integration and system testing of the entire system after the developers are finished implementing.
- *Project manager*: Responsible of the quality of the final system, dividing tasks across the team members, and coordinate them

Critique of the Waterfall model

Even Royce himself stated that the model could be hard to accomplish, and that it could fail in practice (Royce (1987)). The model has also been found to have its drawbacks in the later, where Sommerville (2010) claims that the model fails, as it does not consider the changing environment that the software is being developed within. Also the required documentation was criticised by Sommerville, as he suggests that the focus on documentation leads to too little time being used on actually developing the software. The agile methodology that you will be introduced for below, was developed because of the drawbacks of the Waterfall model, were it is claimed that the Waterfall model gives a overhead in costs for small to medium sized businesses Beck et al. (2001).

2.3 Agile software development

The waterfall model has been found to be troublesome, especially when requirements are changing. The dissatisfaction with this model has resulted in an *agile approach to software development*. Several software developers contributed to inventing the agile approach, and that is reflected upon in the Agile Manifesto. The manifesto states the following priority order:

> *"Individuals and interactions over processes and tools. Working software over comprehensive documentation. Customer collaboration over contract negotiation. Responding to change over following a plan."* (Beck et al. (2001))

In the later several scientists have found weaknesses with the definition of agile introduced by the agile manifesto. Conboy and Fitzgerald (2004) is some of them, and through their research they redefines agility as:

"The continual readiness of an entity to rapidly or inherently, proactively or reactively, embrace change, through high quality, simplistic, economical components and relationships with its environment"

A common trait with all agile methods is that they use an *incremental* approach in order to deliver working software systems. These methods are designed to deliver working software quickly, with no more documentation than necessary, and keep bureaucracy low (Sommerville (2010)).

2.3.1 Scrum

Scrum is an agile methodology evolved from the Agile Manifesto. In Version One's "11th Annual State of Agile Report" this methodology was found to be the most used agile methodology of today. Actually, the total of 84% of the attendees said that they used Scrum or Scrum-like methods (VersionOne (2017)). This source might not indicate the exact number of people using the methodology, as the people who do not use it might not answer the survey. However, it does indicate that there is a relatively large number of people who operate with Scrum.

Scrum was originally developed for small teams, often only consisting of 5 - 10 employees. The methodology keeps the values from the agile manifesto by focusing on delivering *increments* of high value in fixed length *iterations*, called *sprints* (Rising and Janoff (2000)).

The phases of the methodology are illustrated in Figure 2.2 and mainly consist of three phases: *an initial planning phase, sprint cycles* and a *project closure phase* (Sommerville (2010)). The three phases concern the following:

1. The first phase involves getting an overview of the project and designing the architecture.
2. The second phase consists of the iterative sprints, where each sprint calls for four ceremonials: *sprint planning, daily stand up meetings, sprint demo* and *sprint retrospective*.
3. Finally, the last phase includes project closure and product release.

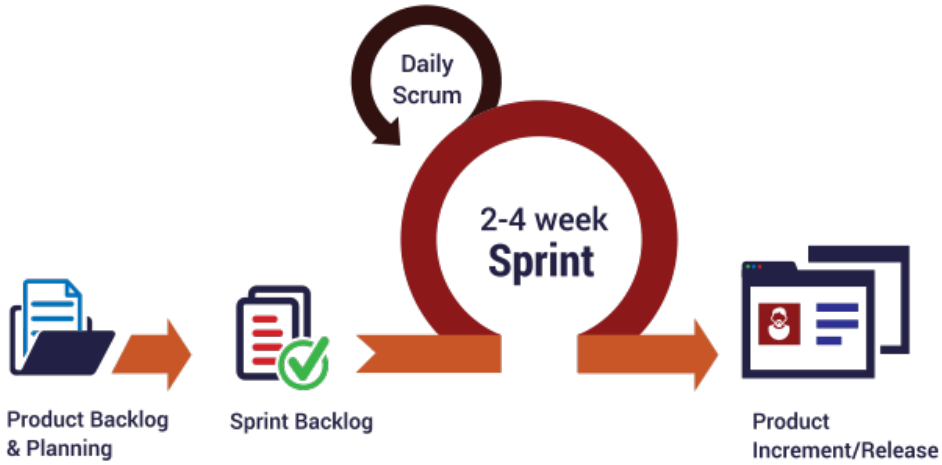

Figure 2.2: Three phases of scrum[3]

[3]https://www.intelegain.com/scrum/, Accessed: 15.01.18

Product and sprint backlog

The first phase of Scrum is devoted to constructing the *product backlog*. The backlog is a priority list, which holds all the tasks that have been identified. In order for the tasks to easily be developed, they shall hold a short description specifying the work required in order for the task to be fulfilled. For example could a task be to develop a profile page. Then then description would be how the profile page shall be constructed, maybe it is required that the profile shows a specific type of information, that the profile is dependent on the log-in information, or that it only shall be accessed if the person is logged in.

In addition to this product backlog, a *sprint backlog* is being constructed in the first ceremonial of each sprint, the *sprint-planning meeting*. The backlog holds all the tasks that shall be finished at the end of the print. Equally to the product backlog, the sprint backlog is also prioritised, but the tasks that it contains are selected from the product backlog (Rising and Janoff (2000)). For example, the profile-task could first be made for the product backlog, then the team decides together with the costumer that the task shall be finished after the up-coming sprint. Then the task is selected from the product backlog and put into the sprint backlog.

Roles of Scrum

In contrast to the Waterfall model, Scrum states that there shall be no project leader in order to hold back on bureaucracy. However, the Scrum methodology allows for several other roles, where they are listed in Table 2.2 and illustrated in Figure 2.3. One of the roles that shall be given to a team member is the role of being a *project architect*. The person selected for this role has the responsibility of ensuring that the vision of the project is consistent with the architecture of the system ((Rising and Janoff (2000)).

There shall also be selected a *Scrum Master* (SM) for the project. The role of the SM is to function as a motivator, guard the team from interrupts and external requests, perform changes in order to maximise productivity, train the team, remove obstacles for the team members, ensure efficient communication and that the Scrum methodology events take place[4].

Lastly, the Scrum methodology allows for a *Product Owner* (PO). The PO has the responsibility of the product's business value and the tasks that is still in the Scrum backlog (Schwaber (2007)).

[4]http://www.shaolintiger.com/2015/07/16/learnings-from-scrum-scrum-master-certification-in-malaysia-psm/ Accessed: 15.01.18

Table 2.2: Roles of Scrum (Rising and Janoff (2000), Schwaber (2007))

Role	Description
Architect	Responsible of ensuring that the vision of the project is consistent with the architecture of the system
Scrum master	Shall function as a motivator, guard the team from interrupts and external requests, perform changes in order to maximise productivity, train the team, remove obstacles for the team members, ensure efficient communication and that the Scrum methodology events take place
Product owner	Responsible of the product's business value and the tasks that is still in the Scrum backlog

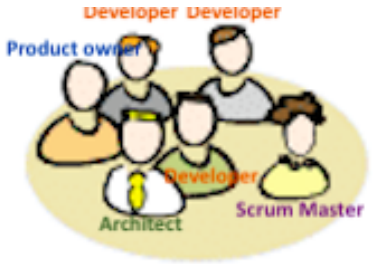

Figure 2.3: Roles of Scrum [5]

Ceremonials of the sprints

In order to keep the scrum team up to date the iterative sprint cycles Scrum call for several ceremonials that shall be done. One of these is the *daily stand-up meeting*, which involves keeping the team members updated on their current tasks, checking if any team members are having any issues, and if any team members' work is affecting other team members' work. If there are any issues or dependencies, this meeting shall bring clarity and promote team members helping each other. These meetings are therefore also providing team-building actions. These meetings shall however, take no longer than 15 minutes (Rising and Janoff (2000)).

Another important ceremonial event is the sprint retrospectively taking place at the end of each sprint. During this ceremonial process the stakeholders shall be involved in order to decide if they shall add any further work, eliminate work or re-prioritise it. This ceremonial meeting shall also be used to evaluate the work they have done and add team spirit. The evaluation shall question

[5] https://www.microtool.de/en/what-is-the-scrum-of-scrums/ Accessed: 01.05.18

what went well and if there are any possible improvements for the *next sprint* (Rising and Janoff (2000)).

Critique of Scrum

Even though Scrum sounds good in theory, there has been reported research showing that when Scrum is used in practice, the methodology tends to be altered to fit the project. The reason for such adjustments seems pragmatic since people using Scrum may feel restrained by the ceremonials that have to take place, and having to do them only because one obligatory *shall* (Dikert et al. (2016)).

The same research also stated that several teams used Scrum as a starting point, but later adjusted it to a more flexible version. This version is meant to fit the project and the team members better. The adjustments are performed on the basic assumption that one shall be careful forcing tasks that seem meaningless on team members. Being obliged to meaningless tasks may lead to frustration and in turn into decreased productivity in the long run (Dikert et al. (2016)).

2.4 Large-scale in agile software development

Having looked at how agile methodology works for small projects, it is appropriate to introduce agile software development in large-scale projects. However, prior to describing how agile methodologies can be scaled up, it is helpful to have an understanding of what large in large-scale means in an agile software-development context.

2.4.1 Taxonomy of large-scale

Until now, there has been found no clear-cut consensus among researcher or software developers on how *large-scale* in terms of software development shall be defined. However, several authors have tried to define the meaning of large-scale. It has been suggested that its meaning includes measures in terms of project duration, the cost of the project, or the number of teams or people involved in the project. Schnitter and Mackert (2011) suggested a description that involved a measure in terms of number of people involved in the project. In their research they stated that the maximum number of people that can be coordinated when developing software agile, is about 130 people.

Other researchers, Dingsøyr, Fægri and Itkonen (2014) wanted to find a suitable taxonomy of scale for agile software development (See Table 2.3, showing the taxonomy of scale). In their research they suggested that a project is of *large-scale* if it consists of 2 - 9 teams. Further on, their taxonomy suggests that a project of *small-scale* only consists of one team, and that a project shall be considered as *very large-scaled* if it consists of 10 or more teams. The taxonomy is later cited by several others, where Dingsøyr et. al (2017) used their own taxonomy in a research of a project that was considered as very large. This taxonomy of scale will be the definition used for small-scale, large-scale and very large-scale in terms of agile software development throughout this thesis.

Table 2.3: The taxonomy of scale (Dingsøyr et al. (2014))

Size	Description
Small-scale	Projects consisting of one team
Large-scale	Projects consisting of 2 - 9 teams
Very large-scale	Projects consisting of more than 10 teams

2.4.2 Scaling Agile

In the chapter above it is outlined how small-scale agile software development take place and the taxonomy of the scale. In this section, I will explain how *agile* software development is scaled up, which leads us to the introduction of Scrum-of-Scrum and a *hybrid* approach to software development.

Scrum-of-Scrum (Large-Scale Scrum (LeSS)

The Scrum methodology, as explained in Section 2.3.1, is meant and developed for small-scale projects. Despite the intentions of the methodology, the interest of using Scrum in larger projects has increased. One result of this increased interest is the Scrum-of-Scrum (SoS) practice, where its structure is illustrated in Figure 2.4. This practice has evolved with the intention of making it possible to use a Scrum-like methodology in large-scale projects (Paasivaara et al. (2012)).

Different from Scrum, the SoS introduces SoS meetings. These meetings are much alike the daily Scrum meetings mentioned earlier. But in addition to daily meetings within each team, the SoS

practices are introducing meetings where one representative from each team has to meet with the representatives from the other teams (See the second level in Figur 2.4.2). Through this inter-team SoS meeting, this practice intends contributes to coordinate and synchronise the teams. The team member, who is selected to represent the team, and attend the SoS meeting, shall in terms of SoS rotate. The rotation is important as the team member representing the team, shall be the person who is best positioned to understand and respond to the subjects to be discussed (Paasivaara et al. (2012)). These meetings shall in terms of Larman (2010) last for no longer than normal Scrum meetings, meaning no longer than 15 minutes. It is therefore important that the meetings get used for what they are intended to, and not only as an arena for handing over status reports to the manager (Larman and Vodde (2010)).

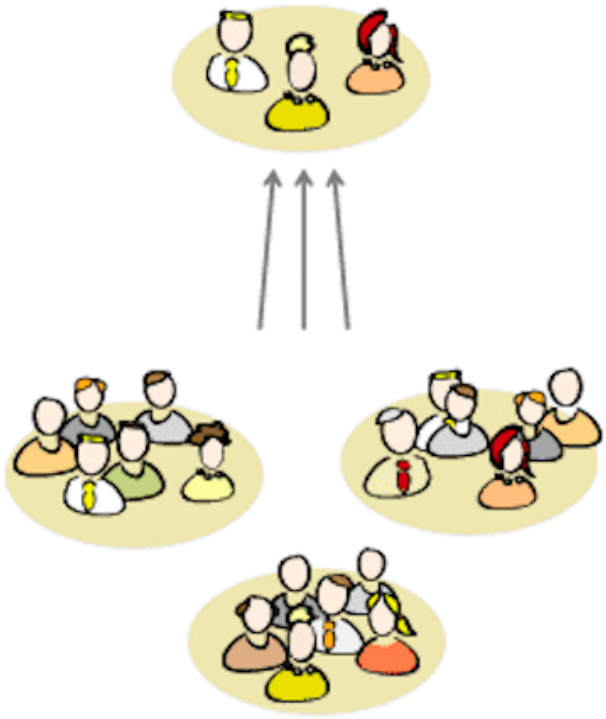

Figure 2.4: Scrum-of-Scrum structure[6]

When projects get scaled even further, resulting in many teams and also very large SoS meetings, the meetings are said to be less effective. In these projects *nested scrum meetings* have been suggested as a solution by Mike Cohnfootnotehttps://www.scrumalliance.org/community/articles/2007/may/advice-on-conducting-the-scrum-of-scrums-meeting Accessed: 25.01.18. This structure results in a Scrum-of-Scrum-of-Scrum (SoSoS) structure as illustrated in Figure 2.5. This structure has been less researched, and the article introducing this structure does not state if the structure works in practice

[6]https://www.microtool.de/en/what-is-the-scrum-of-scrums/ Accessed: 01.05.18

or not.

Cohn is however, not the only one suggesting this structure. Ken Schwaber suggested through his book "The Enterprise and Scrum" (2007) a structure with meetings at several levels that could be used in large organisations. Schwaber conducted a multiple case study involving 14 teams distributed across four time zones. In this research the project had been scaled from having daily scrum meetings to a two-level nested structure. The organisation that was studied arranged regular Scrum meetings, where they had evolved to a two level structure, resulting in SoS meetings. These SoS meeting was led by a senior scrum master, and this person further reported the status in a project SoSoS meeting. In the study, Schwaber reported that the biggest problem was how information was handled across the structure.

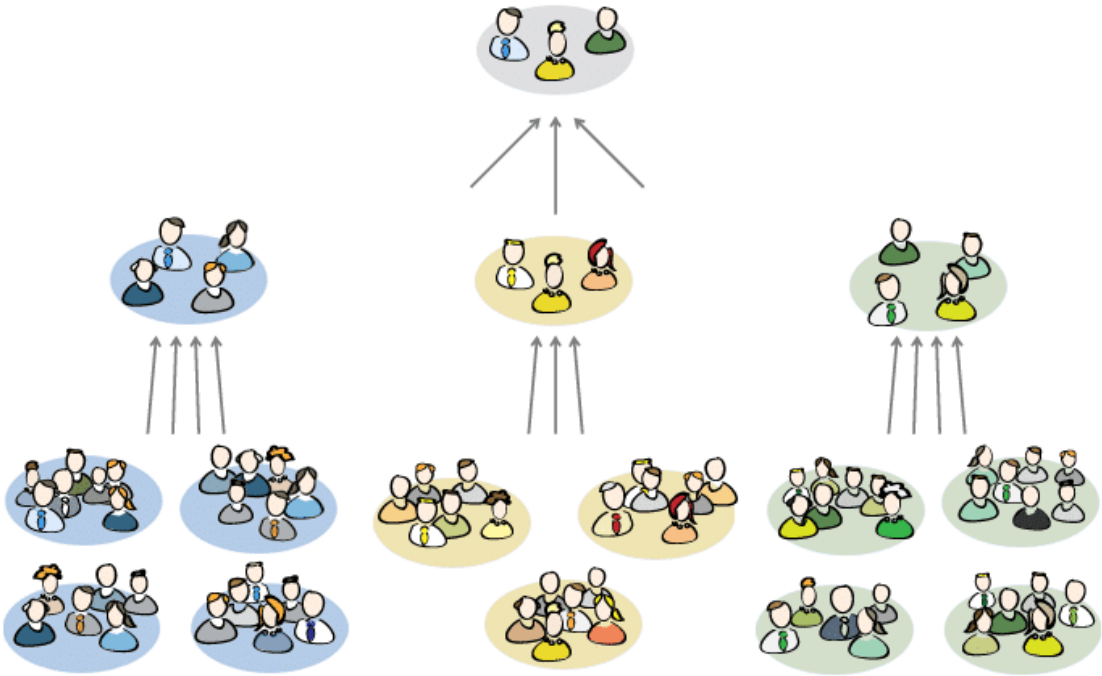

Figure 2.5: Scrum-of-Scrum-of-Scrum structure[7]

2.4.3 Hybrid approach

Because of the challenges regarding scaling agile, hybrid approaches towards software development have gained increased attention. The intention of the hybrid approach is to combine agile and traditional plan-driven methodologies (Batra et al. (2010)). By definition is hybrid "something

[7]https://www.microtool.de/en/what-is-the-scrum-of-scrums/ Accessed: 01.05.18

of mixed character" or "a thing made by combining two different elements"[8]. This implies that the combination of agile and traditional methodologies include a variety of different combinations of these.

Barlow et al. (Barlow et al. (2011)) have through their research suggested a combination of the mentioned methodologies as a hybrid methodology. Through this research they found three key characteristics of a hybrid approach. These findings include that characteristics, practices and roles shall be adopted from two or more methodologies. A hybrid approach will therefore not only be a single type of method, but instead a combination of a variety of mixes of different methodologies. This combination can span from purely traditional to purely agile on each extreme. The research of Barlow will also be the basis for the understanding of a hybrid approach throughout this thesis.

[8]https://en.oxforddictionaries.com/definition/hybrid, Accessed: 25.01.18

Part III

Coordination

Chapter 3

Coordination

3.1 Introduction to coordination

Coordination is a word frequently used in the daily speech, but still many find it difficult to define. This difficulty is understandable, as there exist many definitions of the term coordination, and just as with the definition of large-scale, there has been found little consensus among researchers or software developers on how coordination shall be formally defined in this context. Malone (1988) did an early attempt on defining coordination by the definition:

> *"When multiple actors pursue goals together they have to do things to organise themselves that a single actor pursuing the same goals would not have to do"* (Malone (1988)).

Malone did further research on coordination and together with Crowston they redefined the definition as:

> *"Coordination is the managing of dependencies"* (Malone and Crowston (1994))

They stated that there was a dependency to be managed by coordination when there was a relationship between two or more tasks, and where a task had to be paused because of their connection. There will however, in terms of Malone and Crowstone, be no need for coordination if there are no dependencies. Hench there will be a larger need for coordination as the number of dependencies increases. The meaning of this definition originates from the interdependent relationship between activities, and in order to handle these efficiently, one will have to use coordination mechanisms (Deng et al. (2008)).

Strode (1994) criticised the research of Malone and Crowstone to only be a theory, and that it could

not be used to predict outcomes or coordination effectiveness. However, the research has been found helpful in the later years when identifying dependencies, categorising them and in finding coordination mechanisms (Malone and Crowston (1994)). Another researcher, Osifo (2012), also researched coordination. In his research he classifies coordination as being an element in an organisation. He further states that if there are no dependencies present, then there are no need for coordination. This statement supports Malone and Crowstone's research. Coordination is in terms of Osifo important and necessary for organisations in order to accomplish cooperation through participation and transparency, and in order to set rules and standards internally. In addition, it is important externally in order to define boundaries, to keep focus and establish the right versions of the project that shall be coordinated. Through his research,*"Organisation and coordination"*, Osifo summarises the role of coordination as following:

> *"Coordination is a part of planning, because it tells what to include in a good plan and how to execute it. Coordination is part of organising because it takes the first lead. Coordination is part of staffing, because it specifies who will be a staff and the relational placement. Coordination is part of directing, because it gives a clear focus. Coordination is coordinating. Coordination is a part of reporting, because it makes it realistic. Finally, coordination is part of budgeting, because it gives it a good appraisal"*(Osifo (2012)).

Through this formulation it is clear that coordination can be found in close to every part of a project. It is therefore important to pay attention to how a project is coordinated, not only in terms of plans but also in actual action, as inadequate coordination could slow down the project. Because coordination regards dependencies, it is often also complex. Coordination is therefore often achieved through several mechanisms and not only one, where the mechanisms together help coordinating the project (Dietrich et al. (2013)).

3.1.1 Coordination in large-scale projects

Coordination in large-scale software development projects have been found to be challenging, and especially the coordination between teams. When projects increase in size, the complexity and number of dependencies also tend to increase, which in turn also increase the need for coordination (Kraut and Streeter (1995)). Because of this increased complexity, the use of a team of teams setup has been increasingly practised. And as projects increase even further, resulting in many teams, the coordination of the teams often ends up with a hierarchical setup of team of teams. This setup is in terms of organisational theory defined as a multi-team system (MTS). This

new structure has paved the way for a different handling and challenges with coordination than what has been practised earlier (Scheerer and Kude (2014)). Mathieu et al. (2001) defined MTS as the following:

> *"Two or more teams that interface directly and interdependently in response to environmental contingencies towards the accomplishment of collective goals. MTS boundaries are defined by virtue of the fact that all teams within the system, while pursuing different proximate goals, share at least one common distal goal; and in doing so exhibit input, process, and outcome interdependence with at least one other team in the system* (Mathieu et al. (2001)).

In software development projects of large scale, projects tends to use a similar way of organising, the structure in these projects are often referred to as a Scrum of-Scrum setup, which is the structure introduced in the above Section 2.4.2 (Scheerer and Kude (2014), Larman and Vodde (2010)).

Even though coordination tends to have an even greater role in larger projects, it can be quite complex. When agile software development projects get large, its dependencies tend to increase and get more complex. The need for inter-team coordination is therefore even larger. The role of coordination is important in order to keep the work-flow continuous, and in order to make sure that each team fulfils its tasks within time (Larman and Vodde (2010), Paasivaara et al. (2012), Scheerer and Kude (2014)).

In a study conducted by Bick et al. (2016), they followed five scaled agile software development projects, with five different structures. One of the findings from the research was that there were found several ways in how to do inter-team coordination in large-scale agile software development projects. They also found that the coordination approaches varied in terms of the nature of the coordination.

Of the five investigated projects by Bick et al. (2016) four of them had team members who were overall satisfied with the coordination mechanisms used. The one project who expressed dissatisfaction with how coordination was handled, stated that one of the problems was how communication was handled. As there was close to no communication between teams, all communication took place through a central team. Another problem was how dependencies was handled leading to communication and work bottlenecks. Where teams were left waiting for other teams to be finished.

Other researchers who have studied large-scaled agile software development also found that one

of the problems with coordination was the problem of coordinating dependencies and communication across teams (Paasivaara et al. (2012), Bick et al. (2008), Dingsøyr et al. (2017)). This is a problem that has been noticed by Curtis et al. (1988) as well. They investigated large projects who used the Waterfall model and stated that coordination in large projects can be quite complex, and as a result getting communication bottlenecks and breakdowns are more likely. Kraut and Streeter (1995) found that the number of uncertainties also tends to increase as the projects get larger, and the likelihood of changing the requirements of the project increases. Also they where investigating a project that used the Waterfall model. They found that the change of requirements causes uncertainty in the tasks that shall be done and the software itself. This will further affect the coordination of the whole project, as there will be uncertainty in which tasks that shall be done, the task prioritisation and how the tasks shall be coordinated between the teams. Dingsøyr et al. (2017), on the other hand, investigated a agile software development project that was considered as of very large-scale. Through their research they found that the number of arenas used for coordination tended to increase with the size and complexity of the project. The need for several coordination mechanisms in large projects is also supported by Dietrich (2013). In addition, Dingsøyr et al. (2017) found that the arenas tended to change throughout the project, and some arenas came and some disappeared depending on the needs for coordination at the time in the project. Also, the formality of the arenas was found to become more informal as the project evolved.

Not only is inter-team coordination important in order to avoid bottlenecks, it is also important in order to keep the agility in the project. Melo et al. (2013) identified that the agile team management is the most influential factor for the productivity of the agile teams. Through their case study they also found that inter-team coordination was an inter-team management issue, and that the management of the teams influenced the productivity of the teams. Their findings show that when there is a lack of commitment by one team, this can end up delaying other teams and the whole project. Also if the rules of coordination across the teams are too strict, this will reduce the agility.

Several scientists agree on that there are challenges when scaling agile software development projects, where coordination is one of them. With the large consequences when coordination of teams goes wrong, I believe that the inter-team coordination needs more attention and should be investigated further.

3.2 The delivery model

Already in 1980, Mintzberg (1989) released his study of coordination wherein he proposed five coordination mechanisms. It is important to notify that this study was on coordination in general and not in terms of software development. The coordination mechanisms he identified were the following (Mintzberg (1989)):

1. *Mutual adjustment*: Coordination is achieved by informal communication between two to many individuals (E.g. A person having the a conversation with one or more people)
2. *Direct supervision*: Coordination is done by one person taking the responsibility of coordinating the others (E.g. A Scrum master being responsible for arranging daily stand-up meetings)
3. *Standardisation of work processes*: Coordination is done by specifying or programming the content of the work (E.g. The solution description structured by an architect)
4. *Standardisation of output*: Coordination by specifying the result of the work (E.g. A test that is written that the program has to pass)
5. *Standardisation of skills and knowledge*: Coordination by specifying the required training of the worker in order to perform a specific task (E.g. Requirements for becoming an architect)

Some years later Strode et al. (2012) released their research and presented insight to coordination mechanisms directly related to agile software development, though only in small-scale. The coordination strategy presented in this research concerned synchronisation, structure and boundary spanning. These mechanisms were found to be valuable in order to coordinate effectively. Further, their research used Mintzbergs study and stated that mutual adjustment is important in an agile perspective when coordinating at group level. However, at individual level, coordination is stated to be effective when obtained through one-to-one communication (Strode et al. (2012)).

The scientists cited above have made significant findings in terms of coordination and the use of coordination mechanisms, but none of them have considered coordination at *team level* in their strategy. However, Van de Ven (1976) is a researcher who also considered coordination at team level. This expansion involves mutual adjustment between teams who normally are co-located. The strategy described by the author has many similarities with what Thompson (1967) found almost ten years earlier. Thompson divided coordination into three categories based on the type of dependencies it concerned. The three categories he presented were:

- *Pooled inter-dependencies*: Organisational units performs separate functions. The units does not interact directly nor depend on each other, but are contributing in order to reach the organisational goals, creating an implicit dependency to the other units. This is best coordinated by standardisation, little communication and decision effort.
- *Sequential inter-dependencies*: When one organisational units output is the next units input, where one unit is dependent on the other units work. This is best coordinated by planning, medium communication and decision effort.
- *Reciprocal inter-dependencies*: One organisational units output is also here becoming the input for the next unit. However this model can be cyclical, meaning that one units output can go back to one unit which have already been working on the product earlier in the process. This is best coordinated by feedback an mutual adjustment.

Similar to Thompson, the Van de Ven model of coordination contains the team aspect and inter-team coordination. Van de Ven et al. (1976) also differentiated coordination into three with reference to the three *modes for coordinating work activities*, and they are named: *impersonal, personal*, and *group mode of coordination*. Before describing them further in the sections below, we will take a look at what factors that was proposed by Van de Ven that could predict variations in the three modes. These factors have been recognised by Van de Ven as: *Task uncertainty, task inter-dependence* and *size of work unit*. These factors are also the determinants for which coordination mechanisms and mode that shall be used. The factors are further described in Table 3.1 below. Through their research it was found that a higher task uncertainty lead to an increased use of mutual adjustment through horizontal channels and group meetings. Further, they found that higher task interdependence, led to an increased use of coordination mechanisms across all the three modes mentioned.

Van de Ven et al. also present some observations in their study, which involved the consequences of increasing the size of work units. These findings also concern principles in common with scaled agile software development methods, and are therefore important to highlight. The observations were the following (Van de Ven et al. (1976)):

1. When group cohesiveness decreases, then there is an increase in sub-group formations.
2. The participation of individual members tend to decrease, and use mechanical methods when giving information. In addition they tend to use more direct attempts in order to control each individual behaviour.
3. The management tend to use more impersonal techniques when coordinating rather than

Table 3.1: The three factors deciding which mode that shall be used for coordinating activities (Van de Ven et al. (1976))

Factor	Description
Task uncertainty	Is the measure of how predictable the methods used are, and also the difficulty and variability of the work. Other measures are the complexity of the search process; the time it takes to think through a solution of the problems; the extend to which a process or intervention has predictable outcome; and the time taken before the outcome is known.
Task interdependence	Is the degree of how dependent a task is of other tasks, and if it is possible to divide the work into individual tasks.
Size of work unit	Is the number of people employed to work with the project.

face-to-face.

4. The leaders tasks often increase in numbers and gets more complex. In addition the employees tends to be more tolerant towards more directive and structured leadership.

The focus of Van de Ven's research is on coordination of teams and how the different modes change depending on the factors mentioned. It is stated that coordination of large-scale agile software development is complex. If we translate it to Van de Ven's model, we also know that coordination of a project is rarely done with only one coordination mechanism, and that the task of coordinating a project changes throughout the project.

The next sections describe the three modes of coordination that was mentioned earlier. The modes based on Van de Ven et al. (1976), and other researches that have researched Van de Ven's model in the later years have been included. This is in order to give a broader description and perspective on implications of the research conducted by Van de Ven.

3.2.1 Impersonal mode of coordination

Impersonal mode of coordination involves all coordination types related to programming, administrative coordination and technical tools. Examples of these are plans, rules and hierarchies (Van de Ven et al. (1976),Boos et al. (2011)).also supports this and states that the combination of large-scale, uncertainty and interdependence require specific coordination techniques. The

tools mentioned by Kraut are technical tools, modularisation and formal procedures. Also Strode (2012) found in his study a need for structure when handling coordination efficiently in small-scale agile software development projects. Where they defined structure as; *"the arrangement of and relations between the different parts of a complex structure"*. They further defined structure of concerning how proximity, availability and substitutability, where their description is listed in Table 3.2. Strode's structure can therefore be compared to mechanisms used in impersonal mode of coordination.

Table 3.2: Stode et al. (2012) definition of structure, where structure consists of the elements: Proximity, availability and substitutability

Element	Description
Proximity	Refers to how physically close the other team members are located
Availability	Refers to how available the other team members are for interaction
Substitutability	Refers to the ability to other team members to perform the same task as an other team member

This mode of coordination is intended to help managing issues regarding coordination in large projects. When projects get larger and more complex, this mode is also found to be more important. One of the mechanisms that was noticed by Mintzberg (1989), as introduced earlier, was standardisation. Where he presents three aspects of standardisation: *Standardisation of work, standardisation of output* and *standardisation of skills and knowledge*. This mechanism substantiates the impersonal mode of Van de Ven et al.

The necessity of artefacts that keeps the project organised and coordinated have also been found by Malone and Crowstone (1994). As introduced earlier in the chapter, they found the need for a theoretical modelling framework in order to analyse complex coordination processes. In addition, they also found benefits for using this framework for examination of group work and action (Crowston et al. (2006)). However, Strode (2012) argued this theory to be a less suitable framework to measure *coordination effectiveness* and she stated that coordination mechanisms couldn't be generalised. Despite this, she implied that the theory could be used in order to gain a better understanding of how factors can support coordination.

3.2.2 Personal mode of coordination

In terms of Van de Ven (1976) *personal mode of coordination* involves feedback or mutual adjustment. The individual co-worker achieves mutual task adjustments by giving information through horizontal or vertical channels in the organisational hierarchy. Horizontal channels involve communication between individuals, where they are having one-to-one communication, either across teams or within a team. In contrast, the vertical channels involve communication across hierarchies.

Mintzberg (1989) also introduced mechanisms that substantiate the horizontal and vertical communication. The two coordination mechanisms suggested are: *Mutual adjustment* and *direct supervision*. Another group of scientists who have performed research on coordination theory in the later years is Espinosa et al. (2010). They presented organic coordination, which has similarities to this mode of coordination and to Mintzberg. Organic coordination involves coordination archived through feedback or mutual adjustment. The communication used in order to coordinate could be either informal or formal.

Personal mode of coordination includes both formal and informal communication. This type of communication has also been found to be important for coordination by Kraut et al.(1995). They stated that this communication type is important for both team members and the project. The mix of informal and formal communication brings value by increasing coordination through sharing information. They also suggested that uncertainties are solved through the informal communication across units. Further, Boos et al. (2011) suggest that the personal mode of coordination is useful if the project is not fully scheduled and anticipated. In their study they state that communication between the team members is a dependent factor for personal mode of coordination. This statement is also supported by Dickinson and McIntyre (1997).

One aspect of personal mode of coordination that has been emphasised by several scientists in recent years is trust. The concept of trust has been found vital for progress when projects increase in complexity. However, trust has also been found harder to achieve as the complexity of the project increases (Lehtimaki (1996)). Trust and trusting the decisions taken during the whole project by others, has also been stated as an important aspect of agile methods (Moore and Spens (2008)). The lack of trust has correspondingly been found to be correlated with poor coordination by Smith and Schwegler (2010). Osifo (2012) noted that trust is a part of performance just as it forms basis for coordination.

3.2.3 Group mode of coordination

Just as with personal mode, group mode of coordination also takes place through feedback and mutual adjustment. There is however, a difference in the arenas that the feedback and mutual adjustments take place, as group mode takes place through scheduled or unscheduled staff or committee meetings with several people attending the meeting. This type of coordination requires more planned communication than the two earlier modes, as it often involves change within the organisation (Van de Ven et al. (1976)).

In terms of large-scaled projects this mode is considered as one of the most important one, as several teams will have to be coordinated. Espinosa (2010) found coordination aspects from group mode that fit well with his cognitive coordination. These aspects include the knowledge actors have about each other, and the tasks the others are working at. Having this knowledge can be beneficial, as meetings can be held more evenly and more focused in terms of updates.

Other scientists have referred to group mode of coordination and why its aspects are important when coordinating large-scaled projects as well. Dietrich et al. (Dietrich et al. (2013)) as introduced earlier, are among them. They introduce three coordination mechanisms in their study, which are; *centralised, decentralised* and *balanced* patterns (See Table 3.3 for description). Further in their study, they presented that the diversity in coordination influenced on several aspect. They listed the following aspects being influenced by diversity in coordination: *"information sharing, work-flow fluency between teams, the efficiency of the project, and learning outcomes"* (Dietrich et al. (2013)). This study also supports the study of Mathieu et al. (2001) by underlining the importance of having several coordination mechanisms when coordinating multi-team systems, and of having several levels of coordination in multi-team systems. Group mode of coordination is therefore considered important in large-scale project.

Another aspect in coordination of large-scale project is the *"managing of dependencies between activities"* " as Malone and Crowstone pointed out through their definition of coordination (Malone and Crowston (1994)). As mentioned above, as projects get larger they are also getting more complex, and this is leading to even more dependencies that need to be coordinated. Melo et al. (2013) even more recently stated that proper coordination enables collaboration between teams, handles dependencies and fulfils the needs of the teams. In this context, group mode of coordination is one way of handling dependencies, as well, because during this mode one can coordinate the dependencies across the different teams and fulfil their needs at the same time.

Strode et al. (2012) as introduced previously, described a strategy to coordinate small-scale ag-

Table 3.3: The three coordination mechanisms introduced by Dietrich et al. (2013)

Mechanism	Description
Centralised coordination	Concerns coordination at group level, such as scheduled and unscheduled meetings when introducing adjustments and change.
Decentralised coordination	Concerns coordination across teams, can be both scheduled and unscheduled meetings between teams.
Balanced coordination	Concerns a combination of the two previous mechanisms, meetings both at group level and team level.

ile software development projects. Even though their research concerned small-scale projects, their findings are considered relevant in larger scale too. The coordination strategy concept they introduced, consisted of three components: *Synchronisation, structure* and *boundary spanning*. Synchronisation is the component that is important in terms of group mode of coordination. This component involves synchronisation activities and artefacts, where the artefacts are the product of the activities. These consist of information given to every team members in order to make them accomplish their work. The activities are intended to bring together all of the different team members at the same time and in the same place. The occasion should have a prearranged purpose, and the intention of the gatherings is to gain a common understanding. This component is important when coordinating across teams and therefore it is relevant for group mode as well.

Summing it all up, group mode of coordination can be said to have a direct impact on inter-team coordination. This mode is therefore important when considering large-scale agile software development projects, as it contributes to a balanced coordination, sharing information, efficiency, fluent work-flow and on completing the product of the project (Dietrich et al. (2013))

3.2.4 The Hypotheses of coordination

The three modes of coordination outlined above taken together with the factors mentioned in Section 3.2 deciding which mode that shall be used for coordinating activities (task uncertainty, task interdependence and size of work unit) made the basis for three hypothesis identified by Van de Ven et al. (1976). The three hypotheses put forward are based on one factor each , and are listed below (Van de Ven et al. (1976)):

- **Hypothesis 1:** Increase in the *degree of task uncertainty* for an organisational unit is associated with
    1. a lower use of the impersonal coordination mode
    2. a greater use of the personal coordination mode
    3. a significantly greater use of the group coordination mode
- **Hypothesis 2:** : Increase in the *work flow interdependence* from independent to sequential and to reciprocal team arrangements will be associated with:
    1. small increase in use of impersonal coordination mechanisms
    2. moderate increases in use of personal coordination mechanisms
    3. large increases in use of group coordination mechanisms
- **Hypothesis 3:** Increase in the *size of work unit* is associated with
    1. a decrease in use of group coordination
    2. an increase in use of personal coordination
    3. a significant increase in use of impersonal coordination mechanisms

As can be outlined from the hypotheses, the relationship between size of work unit and modes of coordination shall be opposite from the relationship that concerns uncertainty and task interdependence. An increase in task uncertainty and task interdependence can lead to an increase in the use of group and personal mode. On the other hand could an increase in size of a work unit lead to an increase in impersonal mode and in personal mode, but a decrease in group mode.

Part IV

Research Method

Chapter 4

Research design and method

This chapter introduces the design and the method of the thesis, including a brief description of the literature review conducted in a preliminary project, before you will be introduced for the research method of the case study.

4.1 Literature review

Prior to the planning of the case study, a review of relevant literature was conducted. This review aimed to provide background information and enabling historical interpretation of the coordination of large-scale agile software development projects, in relation to the research problem the case is intended to address.

4.1.1 Database and key-word selection

The prior literature study was conducted during Fall 2017 ("unpublished" Fredriksen (2017)), meaning that some new research could have been published afterwards and in advance of this master thesis. Because of this, I had to conduct a new review of literature following the same procedure as the prior one. When deciding which databases to use, it was important that the databases contained literature related to computer science, and also a wide span of search results from different journals. Table 4.1 shows an overview of the databases used when searching for literature.

Table 4.1: The databases used

#	Database
1	IEEE
2	Web of Science
3	Scopus
4	ACM
5	Google Scholar

Before beginning to search the databases, the relevant keywords had to be decided in order to structure the search, and get as relevant and focused search results as possible. The keyword should reflect the subjects that concern the thesis, which were mainly issues of inter-team coordination in large-scale agile development projects. Because of the subject being rather new, the need to broaden the search with additional more general keywords was necessary to achieve a larger set of data. The keywords used in the search are shown in Table 4.2.

Table 4.2: Keywords used for database search

#	Keyword
1	Agile Software Development
2	Coordination
3	Inter-team
4	Large-scale

After having applied the key-words with *AND* and *OR* operators, it was necessary to focus the search to only search through literature in the categories of *Computer Science*. This focus was necessary in order to get the most relevant literature.

4.1.2 Snowball sampling strategy

To expand the search even further in order to reduce the risk having excluded any relevant research, a *snowball sampling strategy* was added. This strategy concerns the way new research is selected through already chosen research (Goodman (1961)), and was conducted in two different ways:

1. Several of the searched databases (shown in Table 4.1) also provided snowball sampling, as similar suggested articles were included based on the initial selected article from the search.

Table 4.1 displays the different databases used for the semantic search.

2. Each of the articles and publications selected contain a reference list. This list also works as a snowball sample and supplied the search sample with well-written and relevant papers.

4.1.3 Article selection

Having completed the search both through systematic semantic and snowball-based strategies, it was clear that the articles had variable relevance for the research questions of the thesis. It was therefore appropriate to sort out the articles that were not of sufficiently relevance or of high quality. The exclusion criteria that the articles had to pass in order to be further considered are outlined in Table 4.3.

Table 4.3: Criteria for excluding irrelevant articles

#	Criteria
1	If the research domain is different from computer science, the article shall be excluded
2	The article shall be excluded if it is not an empirical study.
3	The article shall be excluded if it clearly is not within the research area.
4	The article shall be excluded if focus in the article is not regarding software development practises

By skimming through the articles and making the described exclusion, this left me with a more focused sample of relevant literature. Further on, these articles were more in-depth read, and carefully selected based on their relevance for the study. This left me with the literature used in this master thesis.

4.1.4 Use of literature study

Through the literature study I managed to discover, analyse and evaluate the literature that had been written earlier with the aim of providing background information and enabling historical interpretation of the subject of analysis.

The sources found through snowball sampling and semantic search resulted in a total of 49 sources of relevance for the project thesis. Of the articles found only *four* articles were empirical re-

searches concerning inter-team coordination in large-scale agile software development projects, which included a detailed description of their method and the investigated projects.

Through the literature review it was found that coordination and coordination of teams was an important factor for large-scale agile software project to succeed. Despite the findings done through the literature study, the subject is still quite new and the research on this field is scarce which make further studies warranted.

4.2 Research method of the case study

As indicated through the review of the literature, the knowledge of this field is scarce. Because of the scarce resources an exploratory case study was chosen for further research. An exploratory case study is often used when there is a lack of literature on a given topic. Also through focusing on a real-life project, it is possible to identify topics worth investigating further. An exploratory case study can be recognised through its exploration of topics (Oates (2006))

Therefore, to answer the research questions, for this exploratory case study research the purpose will not be to give a final concluding remark. It will rather be to give a descriptive understanding of inter-team coordination in large-scale agile software development projects. An advantage of conducting an exploratory case study is its flexibility and that it is adaptable for change. This type of research can therefore be seen as a interpretive or qualitative research, as it aims to understand and explore the factors of coordination in a given social environment. The factors that an interpretive research was characterised by, include multiple subjective realities, a study of people in their natural social environment, several interpretations and qualitative data analysis (Oates (2006)).

4.2.1 Case selection

This master thesis has been done in collaboration with SINTEF, and the interview material from a large project was made available in this research known as "the Case". The project the Case, is a project that SINTEF has an agreement for use in research purposes, where they started with observations in October 2017 before they, in December 2017, arranged interviews with project team members. These interviews had not been analysed by SINTEF in advance of this thesis, but was given as original research material for this study, as they wanted insight into inter-team

coordination in their project. The Case was an automation project for standardisation of proceedings.

4.2.2 Data collection

In order to achieve a detailed understanding of the case, quantitative data was collected from November 2017 and March 2018. During this time period 12 semi-structured interviews with key informants of the development unit was conducted by SINTEF. The interviews followed three different templates with a focus on architecture, method adjustment and coordination, where all the templates are found in Appendix A.

I conducted observations of the different meetings in March 2018, where I observed the *Scrum-of-Scrum meeting* and a *daily stand-up meeting*. In addition I attended a lecture with the project leader in February 2018. During the observations I took notes and looked at who communicated with whom. During the lecture, I also took notes, asked questions for declarations and got a hand of the presentation afterwards.

For the interviews, since the whole case was located at one site, it was possible to interview all the interviewees face-to-face at the location of the project. The total of 66 employees from the delivering organisation, were working on the development of the software solution and divided into a total of four teams plus a central unit and support units. The interviewees included two *Scrum Masters*, one *tester*, one *functional architect*, three *technical architects*, two *developers* and three *"other"*. Where "other" refers to people with other roles within the investigated case (See Table 4.4 showing the details of the case). The informants were chosen based on their role and their ability to report on coordination on team and inter-team level, architecture and method adjustments. The interviews were later transcribed and anonoymised, also by SINTEF, before it was handed over to me. The recorded interviews consisted of a total of 9 hours and 47 minutes, and were transcribed into 188 pages. These transcribed interviews were then kept safely by encrypting them before saving them on a password protected PC within a password-protected file.

4.2.3 Data analysis

In order to analyse the given interviews thoroughly, an extensive thematic analysis was used. It is a systematically approach used to analyse qualitative information in order to gain knowledge,

Table 4.4: Criteria for excluding irrelevant articles

Category	Description
Product	Software solution for automation of proceedings
Process	Agile
Sites	Norway, Oslo
Number of teams	4
Number of employees	66
Number of interviews	12
Roles of interviewees	Scrum Master (2), Tester (1), Functional Architect (1), Application Architect (3), Developer (2), Other (3)
Total length of interviews	9h 47min
Interview length	0.5-1h
Num. of transcribed pages	188

and is often used when trying to give unrelated data meaning. This analysis was necessary due to the amount of data given, and the need to abstract the themes and patterns of highest relevance for the research. At the same time it gave me the opportunity to get insight necessary to get a deeper understanding of the data given.

The first step of this analysis was to describe the nature of the data, in this case data through interviews of team members of the Case given from SINTEF. Further, the interviews were properly read through, before the coding took place. The coding of the interviews could be conducted with the help of a computer-aided tool or by hand. The method used in this research was the use of the computer-aided tool, NVivo[1].

NVivo

NVivo is a software tool designed for analysing and coding unstructured text used in a qualitative research. The tool is helpful when having to conduct a deep level analysis of large amount of unstructured data such as text including interviews and notes from lecture and observations. NVivo helps in classifying, sorting and organising the text that is given through coding the text into nodes and cases. Coding with the use of NVivo helps in exploring the relationships between data and combining the data by facilitating cross-examination of the text. The tool also facilitates the formation of links, shaping of data and models for further exploration of relationships.

[1]https://www.qsrinternational.com/nvivo/home, Accessed: 19.02.18

Research model

The data given was coded both by hand and into nodes with the help of NVivo for further analysis. In order to categorise that data further, Van de Ven's model of coordination (as described in Section 3.2) has been selected in order to frame the analysis. The reason for choosing this model is because it focuses on *teams* and their *relations*. In addition, the model has been thoroughly researched and combines several coordination theories.

When searching the literature databases, I only found two papers that used the Van de Ven model in their research on inter-team coordination in large-scale agile software development (Dingsøyr et al. (2017), Dingsøyr et al. (2018)). The model has also been used by other researchers when examining coordination in other types of projects earlier (Dietrich et al. (2013), Zmud (1980)).

The use of a predefined model and its factors can lead to valuable new insight into a case. Through this research it might therefore be possible to confirm Van de Ven's model hypotheses of coordination and the relevant influential factors. Or it will reveal vulnerabilities with the model in concern of inter-team coordination in large-scale agile software development projects. The thematic analysis of the interviews revealed several factors from Van de Ven's model. How this was conducted is described in the next section.

Coding analysis

The coding of the interviews was based on the Van de Ven model introduced in Section 3.2. The coding in this research was conducted through the following steps:

1. First, all the interviews were thoroughly read through and notes where taken of interesting findings.
2. Further the use of word search in Microsoft Word was used to search for the different factor introduced by Van de Ven, namely: *task uncertainty, task interdependence* and *unit size* (See Section 3.2). Subjects that related to these factors was also used for the search.
3. In this step the NVivo tool was used. The first coding in NVivo was based on the three modes of coordination by Van de Ven et. al introduced first in Section 3.2, namely: *Group, personal* and *impersonal mode of coordination.*
4. Further, the use of the factors introduced by Van de Ven was used. The second coding was necessary in order to get an even deeper understanding of the data. And as a result of this

step, one could see a relationship between the modes and factors.

5. Other aspects regarding the research question were examined. These included *inter-team coordination* and *agile practices*. The goal of using these keywords was to see if there were any quotes directly related to the Van de Ven model, but also to see if any quotes were missed out.

6. Finally, the coded nodes were analysed, by comparing the nodes and cross checking patterns. This phase included running queries and search through the data. These actions were taken in order to observe connections and patterns between the nodes in comparison with the factors by Van de Ven.

As with thematic analysis, the code shall always be reviewed and validated by more than one person. This is in order to ensure integrity, weather the results are unbiased and to check that the results has not been misinterpreted. As this research is only conducted by one person, this was solved by using the supervisor and ask him for clarifications regarding the coding and discussing the results. The coded nodes were in addition validated several times, and the data re-read.

4.3 Quality and bias to the method

This section includes the aspects relevant to validity of the method and to clarity prior to the analysis. These issues are discussed with respect to the results in the section of limitations.

Due to only one person having conducted the research, and hence having only one person's point of view, the results will most probably be biased. In contrast, it has been shown that having several researchers involved during the analysis helps to reduce such risk of bias (Runeson and Höst (2009)). The validity of this research would therefore be more trustworthy if it had been conducted by more than one researcher.

Further, if this research was to be conducted by another researcher at a later time, the results shall, hypothetically, be replicated (Runeson and Höst (2009)). In this case, however, due to the model used in this research, the results are expected to be similar, but not exactly the same. There are also other factors that might affect the results, such as the interpretation of the model used. There may exist several ways to interpret the Van de Ven model, and in this research one perspective has been used. However, if another researcher aims to use another interpretation with other factors, the results may not turn out the same. In addition other researchers may value and find

other aspects of importance during the coding which also could affect the results. These factors mentioned are also effects of only having one researcher conducting the research.

Rubeson and Höst (2009) also state that internal validity of the study shall be considered when evaluating the quality of the study. Internal validity refers to whether the effects observed in a study can be explained by the independent factors. For this research, evaluation of the internal validity of the analysis mainly refers to the Van de Ven model. However, this does not mean that other approaches will not lead to similar results. A threat to the validity of this case to be considered is if there are influential factors that have not been identified by me through the analysis.

Lastly, in order to verify the theoretical overview that has been gained through the case study, the results and conclusion have been substantiated both by what has been experienced through the analysed case and the theory. If there had not been any time constraints, this could probably have been done more thoroughly.

Part V

Results and Evaluation

Chapter 5

Results

This chapter will present the results given through the analysis of the investigated project. Because the project was found to be quite complex, a description of the project is given before the results of the analysis structured by the use of Van de Ven's model of coordination.

5.1 The investigated project - Its structure and roles

The investigated project started in February 2017 and was estimated to last until March 2019. The project is stated to be one of the biggest software development projects taking place at this time in Norway. The product that is to be developed is a software solution that is intended to automate proceedings. The solution will be developed from scratch, but is dependent on several other old solutions, making the project even more complex. It is expected that more than 100 000 people a year in Norway will use the software.

Further, the project has grown from eight persons and one team at the beginning, to the total of 66 people and four teams involved in the development of the solution from the delivering company (here after referred to as *the Case*) at the time the investigation took place. The number of people also includes a *central unit* and *functional unit*. The central unit includes the project leader and other management roles, and the functional unit includes roles that is considered to ease the work for the teams. The number of people and teams are planned to increase even further, but the teams have to work efficiently before expanding. In addition, the customer has the same amount of people available for the Case to cooperate with, which results in a total of 132 people involved at the time when the interviews took place. The project is expected to require >200 000 man hours

and take about 2 years to finish.

The project is structured in a Scrum-of-Scrum manner with a central unit, and has been structured this way from the beginning. This is despite only starting out with eight people. The teams consist of highly specialised professionals, working as independent from the other teams as possible. In addition, each team has to build expert competence in their domain. All the teams are co-located in one wing of a building, and placed at the same floor as the representatives of the customer. The teams are however, moved around the wing depending on the tasks they are working at and their dependencies.

Even though the size of the teams and number of teams have varied and will vary, they always consisted of one scrum master, three-four developers, one technical architect which can also be a developer and one tester. The description of the roles in the teams is described in Table 5.1 below. One thing one shall notice is that the role of being a scrum master did not rotate as suggested by Paasivaara (2012), but was a specialised role belonging to one person at each team.

Table 5.1: Project team roles and description of them

Role	Description
Scrum master	Has direct contact with project manager and arranges scrum activities within the team
Developer	Is a team member and gets assigned tasks from the project backlog that shall be developed
Technical architect	Is responsible for developing a solution description from a technical perspective, the person is often also a developer
Tester	Has the responsibility of developing test cases and conduct tests at team level

In addition to these roles, you will see that some persons that are referred to as "Other". These persons belong to the functional unit that functions as a supportive unit for the teams. The functional unit includes the solutions manager, other types of architects (functional, security, informational and solution), environment manager, quality securer, customer manager, controller, test data manger, test automation manager, performance test manager, delivery test manager, delivery project support, delivery constructional manager, product owner, and at last there is the project manager. The different roles can be classified into four units, together being organised as a matrix-like structure, with a person, the project manager, being responsible for coordination and organisation of the units. Roles that fall outside the structure are the customer manager, quality

securer, controller and delivery project support. The units are *business, architecture, development* and *test*, where the development unit of the structure is the one that will be focused on in this research:

- *Business:* Responsible for the analysis of needs by defining, prioritising epics, user stories and the project backlog, and delivery description. This unit consisted of a product owner as well as the functional architects and technical architects from the development teams.
- *Architecture:* Responsible for defining and constructing the entire architecture of the system. They are also responsible for describing the user stories and solution descriptions. The unit consists of a delivery solution manager, solution architect, security and information architect and functional manager as well as the technical architects from the development teams.
- *Development:* Responsible for the developing the entire system based on the solution description and user stories given from the architecture unit. Each team consists of the roles defined in Table 5.1.
- *Test:* Responsible for the testing procedures and approving deliverables from the development unit before delivery. The unit consists of delivery test manager, performance test manager, test data manger and test automation manager together with the testers from the development teams.

The illustration of the matrix-like structure is shown in Figure 5.1 below. The business and development units was always considered in the architecture and test planning meetings. The matrix-like structure illustrates a structure where the development teams at the same time as focusing on development also devoted resources to project architecture through their functional architect, and test through their tester. In addition the business unit of the project devoted resources to architecture through solution description, and test through acceptation criteria.

In order to reduce risk of failure in this large-scale software development project, they have divided the project into three main releases. Where the releases are distributed over the two years the project was estimated to last. A expected time-line showing the initiation of the project, when the releases are expected to be finished and the time of the expected completion, is shown in Figure 5.2 below. These releases could also overlap at times and was always under planning, construction and testing, so that no employees were never without work. The roles introduced in Figure 5.1 were therefore constantly working on the construction of release *"n"*, approving deliverables in "n - 1" and analysing needs and constructing delivery description in "n + 1", as illustrated in Figure

5.3. When the approvals were finished, the new release got acceptance tested, put into production and went through an approval phase before being accepted by the customer.

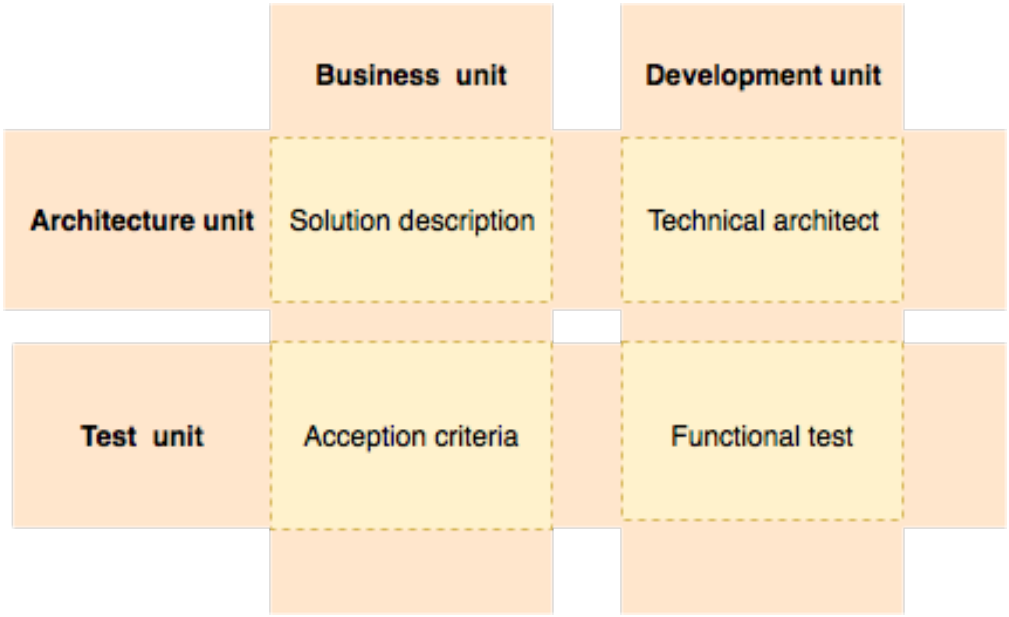

Figure 5.1: The project organisational structure as a matrix

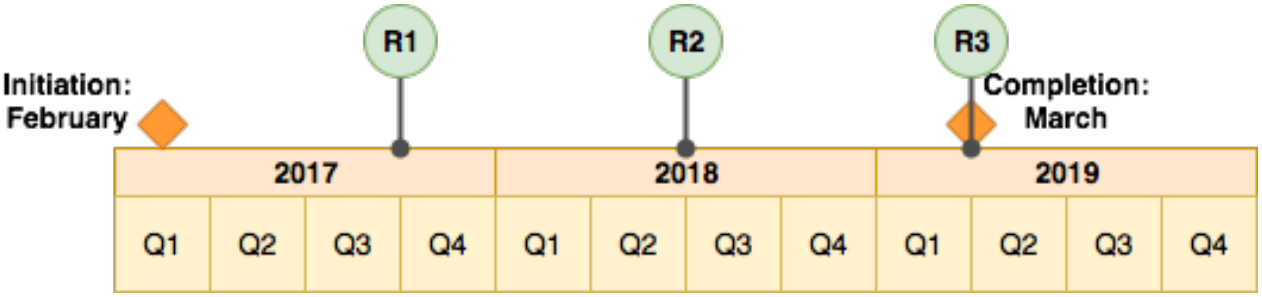

Figure 5.2: The expected time-line of the investigated case. Showing initiation time, expected time for each release and completion time

In the Case they call for several *ceremonies* which promote coordination within teams, across teams and across hierarchies. The ceremonials that takes place on a weekly basis include weekly Scrum-of-Scrum meetings, daily-stand up meetings within teams, the retrospective meeting, weekly technical review, weekly architectural forum and weekly architectural meeting.

The Scrum-of-Scrum meetings in the Case took place once a week, on Tuesdays and Thursdays, where the scrum masters attended as well as the project leader. In addition was the lead and functional architects, and testers invited. The meeting was not intended to take no longer than 15 minutes. Each team arranged daily stand-up meetings, that took place internally in every team

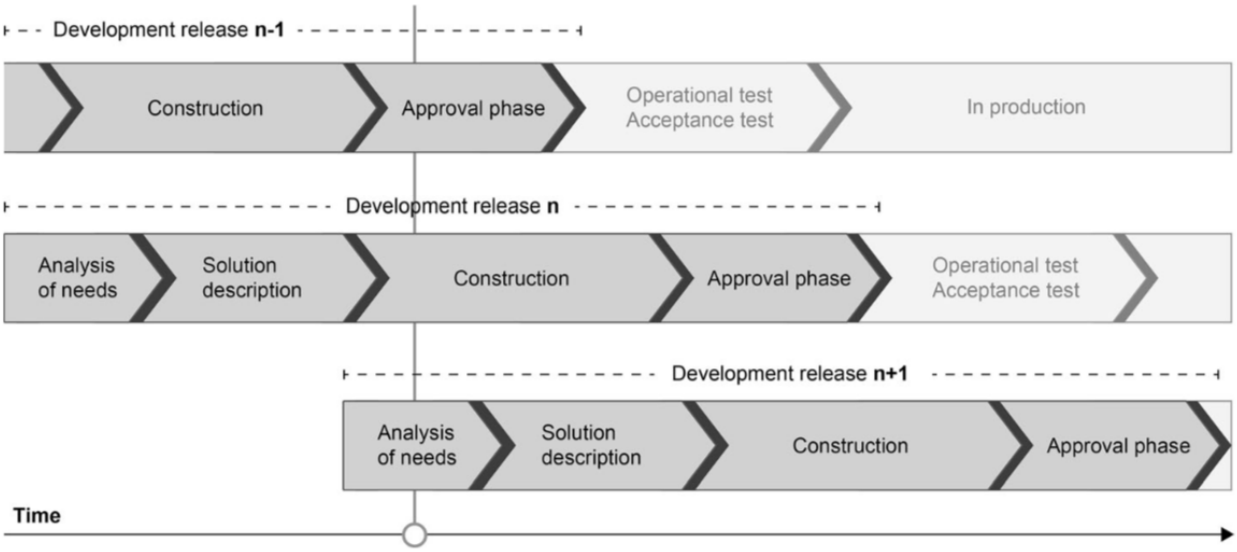

Figure 5.3: The structure of release model (Dingsøyr et al. (2017))

each day, also these meetings were not intended to take no longer than 15 minutes. Further, the teams had technical reviews that took place every two out of three Fridays, where one team was responsible for the meetings, and the team who was responsible was rotating. The third meeting was resigned to the retrospective meeting. Also, the architects had weekly meetings called the architectural forum. This meeting took place every Tuesday, where all the application architects attend together with the lead architect. In addition to his meeting, every Tuesday an architectural meeting took place for the technical architects where they discussed the upcoming user stories. All the weekly meetings are illustrated in a time table in Table 5.2.

Table 5.2: Timetable showing the different meetings

Monday	Tuesday	Wednesday	Thursday	Friday
-	Scrum-of-Scrum	-	Scrum-of-Scrum	Technical review*
-	Architecture forum	-	Architecture meeting	Retrospective**
Stand-up	Stand-up	Stand-up	Stand-up	Stand-up

*The meeting took place every two out of three Friday
**The meeting took place every third Friday instead of Technical review

In addition to these meetings, they also have other coordination arenas and methods, where all of them, including the mentioned meetings, are listen and described in Table 5.3 and Table 5.4 below.

Table 5.3: Coordination arenas and methods

Arena/Method	Description
Stand-up meeting	Was arranged daily by the Scrum master in every Scrum team. The meeting lasted for 15 minutes, and was used as a status update meeting for discussing challenges and progress
Scrum-of-Scrum	Was arranged every Tuesday and Thursday, where all the Scrum masters attended. In addition all the lead and functional architects, and testers was invited
Technical review	This was a meeting that was arranged every two out of three Friday. One team has the responsibility for the meeting every time, where the one with the responsibility circulates. During the meeting they discuss difficulties regarding technology and lessons to be learned for the future
Architecture forum	Is a meeting that took place every Tuesday, where each teams functional architect attended together with the lead architect. During the meeting they discussed dependencies and user stories.
Retrospective	Was arranged every third Friday. Every team focused on a time-line and what they had done during that week
Demo	The demo was arranged at the end of each sprint in order to show the other project members what they had accomplished during the sprint. In addition a larger demo was held at the end of each release for the customer, where a demonstration was conducted
Start-up meeting	A meeting that was arranged in advance of each sprint, on Thursdays at the end of each sprint. The meeting was used to go through the user stories that each team was assigned to. In addition dependencies of each task was introduced and shown on a dependency map
Testers meetings	This meeting was arranged by the testers from each team, where they discussed their test-strategy and if they had any difficulties

Table 5.4: Coordination arenas and methods continued from 5.3

Arena/Method	Description
Architecture meeting	All the teams technical architects meet weekly to discuss technical issues related to user stories
Planning meeting	A meeting that was held within each team at the beginning of each sprint. The focus of the meeting was to plan the sprint and divide tasks.
Acceptation meeting	Meeting between solution managers from the Case and with solution unit from the customer side, where the solution descriptions are presented.
Board	Every team had its own whiteboard showing status on tasks they were working at. The board was also a centre for discussion between team members or across teams when necessary
Jira/Confluence	Tools that was used to keep track on tasks and the project backlog. Jira was used to keep user stories, architectural information and information regarding the project as a whole, status on current sprint, task status ect. While Confluence held information regarding the teams, such as solution description, team routines, checklists, guidelines, retrospectives ect.
Lynk	Is a communication technology that was used as a communication and coordination channel for the testers.
Co-location	All teams, management, supportive functions and representatives from the customer was located at the same floor.
Specialised teams	All teams where specialised on their own domain

From the table it is clear that the project uses several mechanisms in order to coordinate. It was actually found 16 different coordination arenas or methods. The Case had meetings at three different levels, looking much like the Scrum-of-Scrum-of-Scrum structure. This structure is illustrated in Figure 5.4, where one can see the three levels: Team level, Scrum-of-Scrum and Meta scrum. Where Meta scrum is the Start-up meeting where the whole project were gathered. The arrows across the different levels illustrates the informal communication that took place outside the meeting arenas.

5.2 Modes of coordination corresponding to Van de Ven

In this section we will look into the findings from the interviews and observation, and how they fit into the three modes of coordination that Van de Ven suggested, namely the *impersonal, personal*

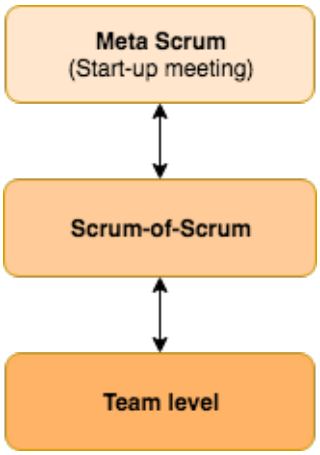

Figure 5.4: The three levels of meetings

and *group mode of coordination.*

5.2.1 Impersonal mode of coordination

Impersonal mode of coordination involves all coordination types related to programming, administrative coordination and technical tools. Examples of these are plans, rules and hierarchies (Van de Ven et al. (1976),Boos et al. (2011)). In terms of Ven de Ven's hypothesis 3 (See Section 3.2.4) the impersonal mode of coordination is expected to increase when a project gets large. This can be explained by the necessity of predefined structures of some parts of the project, which will result in a better understanding of the tasks, and therefore get the project started, making progress easier later, see examples of this type of coordination mentioned in Section 3.2.1. In terms of the Case there has been a use of predefined structures. First of all, they have standardised the solution description of tasks with the use of thorough user stories, with tabs for different types of information:

> *"...When we get the user stories, at least until now, they have been quite detailed, if you can say so. The standard follows the so called "Customer method"-method. Where there is a functional tab, UX tab, information-model tab, architect tab, test tab etc. So on the first delivery they have been filled out quite well. They were very self describing."* - Scrum master 2

Even though it is attempted to work as agile as possible, the Application architect 1, believes it is still necessary to have some standards when it comes to developing the code base and implementation structure. Therefore the user stories were designed and formulated by the architects

in Confluence: *"We are documenting everything in confluence" - Application architect 1.*

In addition all the architectural guidelines for the project were made in advance of the project, and were available for all the team members at Confluence. However, their description were stated to be a bit vague and sometimes the handover between the architects and teams lead to small delays.

> *"The architectural guidelines are very vaguely described in Confluence, but they are very high level or a sort of random, if you can say so"* - Scrum master 2

This vague description resulted in that the team members used the solution description instead most of the time:

> *"...It is the solution description that is our main source, but there are still some architectural drawings that are central and are followed and kept an eye on, but they are also more stable"* - Scrum master 2

The page Confluence was used much more than only a place to hold the architectural description, and was frequently used by the teams. The reason for this was that Confluence holds all the information regarding the different roles involved in the project, the team's routines and guidelines, routines across teams and system documentation, which includes an overview of dependencies. Having a common area online available for everyone, made the coordination easier, as everyone knew where they could find information. The testers for instance, found Confluence quite helpful when trying to involve the team members in testing early:

> *"...it was useful to have a page in Confluence, where I hold a implementation plan that is visible for the whole team. And get the developers a bit more active in the testing as well."* - Tester 1

However, the person responsible for the construction was not as pleased with only having the overview of dependencies at Confluence in the beginning:

> *"One of the difficulties with dependencies is that we only have had it available at Confluence and not in Jira, which is our common work tool"* - Other 1

This brings the topic over to Jira. Jira is also an online arena that they used as a tool for coordination. The arena was used as an online tool for holding tasks that should be done, their status and as their common work tool, and where all the teams had insight into all the teams' tasks, and dependencies as well, since employees requested that. Jira has since then, been the tool that they seek when they need to get an overview of the dependencies:

> *"It is Jira that we use when getting control of dependencies, first of all..."* - Scrum master 2

> *"...It has been more important that the dependencies can be found easily for everyone, that we can find it at Jira in addition, it is because of this easier for everyone to follow, for us who do not sit with the teams..."* - Other 1

The project benefits from using tools like Jira, as it becomes a source for information gathered at one place, that is available at all times for everyone. In addition to Jira, the teams used a board that contained the same tasks as Jira, but that was more visible for everyone to see, also the other teams. This board was kept up to date resulting in little doubt in what tasks the team members was working at and their status:

> *...I think it is fine to get it up on the board as well- That you have it in front of you, but we are all agree up on within the team that it is Jira that is the master, it shall be updated at all times, then I also try to keep the board updated at all times as well... I also believe that the project leaders like to take a quick look at the boards when they pass in the hallway...It also happens that some people discuss in front of the board. That they all of the sudden start to think of a note that hangs there."* - Scrum master 1

Task uncertainty

In terms of *task uncertainty* and impersonal mode of coordination, it was not found many relations, except from the use of solution description. This shall be natural in terms of large scaled projects as the impersonal mode implies more standard methods, planning and schedules. Because of this, if it has been done properly, there shall not be possible to interpret a task in several different ways out of the solution description. There were however some difficulties with the understanding of some tasks and their solution description given by the architect:

> *"...When we are sitting a totally different place we don't really see the dynamics. Then it appears a need for declaration, or they might not see that there is a need for - We had some examples from the previous delivery where the task was solved, without knowing about all the needs of the task..."* - Other 3

Even though there were some uncertainties regarding tasks, they tend to be easily solved by the persons it concerned through asking a functional architect on the team:

> *"It happens in all sprints actually, that one have started to develop, then there is some-*

> *thing one is uncertain about that is added to the solution description that one don't get. But I think it is completely normal, really, that you talk to the functional architect as it appears, I believe it is smart."* - Developer 1

In addition there was found an uncertainty regarding the length of the project backlog and weather there is enough hours to use.

> *"That you are uncertain about if we got enough tasks in the queue for the next iteration, and if you got enough hours to finish the tasks..."* - Other 2

These uncertainties appeared despite using impersonal mode of coordination through their plans, schedules and guidelines. This indicated that it was hard to have plans for everything. Having plans for everything was not intended to either, as one would like to have some slack and agility. Their impersonal mode might therefore not be the best mode of coordination for handling task uncertainty. Having plans, schedules and guidelines might, however, be beneficial in the beginning of the project in order to develop a good base for the project, and train the employees to solve different struggles in a specific way in the future.

Task interdependence

The need for coordination tended to increase as the complexity and the number of dependencies increased. Impersonal mode of coordination calls for having clearly divided tasks and specific goals. The Case used impersonal mode of coordination by having thorough solution description for each task, where each team got their own tasks with their solution descriptions that they were responsible for during the Start-up meeting. In addition, Jira, also held an overview of the dependencies between the different tasks. All the teams also used a physical task board that was always available and visible for everyone. Having these resources available at all times was helpful in order for the teams to have a good overview of the tasks, and work has independent with the tasks as possible.

An additional method used by the Case, was the use of specialised teams. The teams were specialised on their domain, meaning that they got more ownership to tasks related to their domain. In addition the other teams knew exactly who they had to ask or cooperate with when dependencies appeared.

The architects carried out their planning ahead of the Start-up meeting, since they were responsible for developing the solution description and decided when the different tasks shall be worked at. In addition they also operated with a time line. The time line was used to show when the

different tasks shall be worked at in order to coordinated dependencies, so that one task that was dependent on another should have been developed after its dependent task was finished:

> *"...Operate with a time line when coordinating tasks and its dependencies, at least as far as they could predict the dependencies. Where one task that is dependent on an other shall be developed after its dependent task is finished."* - Project leader (In notes from the lecture)

Despite this time line for coordinating dependencies between tasks and the dependency board, there still appeared technical dependencies that were not foreseen. Because of this, it was important that the teams could easily communicate with the other teams to solve these unforeseen dependencies.

This implied that the seating of the teams, and being co-located, were important for the teams to work efficiently. Also the appearance of unforeseen dependencies implies that when the project became larger and more complex, there might be factors, such as dependencies, which are difficult to anticipate and affect the progression of the project even though there was use of plans and schedules.

Overall, coordination was achieved through thorough planning ahead of the Start-up meeting, the always-available overviews of tasks, dependencies, solution description and co-located teams. Which made it easier to have insight into what they should be working at next, both within the teams and across the other team

Size of work unit

With a large-scale agile software development project the complexity is known to increase, and so do the number of people involved as well. This increase in size requires more structure, which implies a larger use of impersonal mode of coordination. From the Case and the quotes highlighted earlier it was clear that the use of coordination tools for communication and in order for everyone to gain project insight was important. In addition several team members highlighted the different meetings as important for communication. However, it was noticed that there was an increased difficulty to have completely overview of everything when the project got large:

> *"...It is clear that it is difficult to have full control, when there are so many people"* - Scrum master 2

The same was mentioned from the project leader during the lecture:

> *"When people ask me if I have full control over everything in the project I often tell them: No, if I would have full control, then the project would have had too slow progress, not being effective at all"* - Project leader (In notes from lecture)

Differently from small-scale agile software development, the Case have seen the importance of having a more hierarchical structure and defined roles, as they used a central unit for coordination. Having defined roles with their own responsibility can lead to single point of failure. The dependency on the main architect was also recognised by Scrum Master 2 to be a potential single point of failure:

> *"He got a large responsibility for the architecture, witch is both good and bad, because it gets a very strong personal dependency to him..."*

In addition they have seen the importance of having arranged meetings at different hierarchical levels which involved team members from all the teams gathered at one place (Such as Scrum-of-Scrums, architectural forum and testers meeting), arranged meetings where all the involved parties were involved (such as the Start-up meeting), as well as meetings at team level.

From the findings one can see that the Case used impersonal mode of coordination at a high degree and benefited from using it. This implies that the use of this mode has been crucial for coordination to function as well as it does. Also when no one can have fully control over the project at all times, this increases the importance of plans, schedules, a coordinating central unit and technical tools.

5.2.2 Personal mode of coordination

Personal mode of coordination involves in terms of Van de Ven (1976) feedback or mutual adjustment. The individual achieves mutual task adjustments by giving information through horizontal or vertical channels in the organisational hierarchy. Horizontal channels involve communication between individuals, where they are having one-to-one communications, either across teams or within a team. In contrast, vertical channels involve communication across hierarchies. See examples of this type of coordination in Section 3.2.2. The three hypothesises by Van de Ven state that the use of personal mode of coordination is expected to increase when the unit size increases and when there is an increase in task uncertainty. Both are often related to large-scaled projects.

In the Case one of the methods or arenas that was found very helpful was the use of informal communication and co-location, which related to both vertical and horizontal personal modes of

coordination. The use of co-location made it easier to reach out to others possessing the right competence, which could be to other team members, and team members in other teams or functional units. It can potentially lead to an extended information flow across teams and hierarchies. The co-location was actually seen as a success factor for the whole project by the construction manager of the system. The person considered informal communication as the most important communication that took place:

> *"So it is a lot of informal communication, which I mean is the most important one and the most effective.* - Other 1

Also one of the scrum masters highlighted co-location as an important factor for success, as it lowers the threshold for speaking to one another:

> *"...The possibility of being able to sit close to one another- It has so much to say. The threshold for talking to one another is so much lover than when you have to walk for a bit.* - Scrum master 1

The same scrum master even stated that by being co-located one could look at the whole project as one team, which could lead to a better environment:

> *"We are really one big team."* - Scrum master 1

Also when the functional units tended to be located at the same place as the teams, they experienced that the team members asked much more questions than when they were located at an area for themselves. In addition some might get their answers from overhearing a discussion by others, or engaging in others discussions:

> *"...it is a huge difference, when they are allowed to sit with the teams they get much more questions than when they sit a whole different place and the teams cannot see you, because when they see you it is much easier for a developer to: "I can probably just ask" or that they overhear any discussions.* - Other 3

Others indirectly appreciated the co-location as well, when they highlighted the ease of just walking over to other scrum masters or team members if there were any questions regarding status on tasks or dependencies that had come up. Then one could easily coordinate with little effort:

> *"It is pretty easy to just figure out of status. Regularly I just ask them directly. Sometimes I also attend the other teams stand up meeting."* - Scrum master 2

This also brings the attentions to the physical boards that they used during the stand-up meetings, and that were always visible for everyone to see. The board was a centre for coordination and

information within the teams, and was related to horizontal personal mode of coordination. For many team members and the project leader, it was important to have an overview of the current status of the tasks. The board covered this need, as it was always available and one could easily glance at the board to see status. In addition the board was used to go through tasks during the stand-up meetings, but also as an area where information was given regarding the project, where the scrum master used the board to clear up any misunderstandings.

> *"We have boards that we use for discussions too, it is based on which type of activity we are having and what is required for that activity. If we can just take it on the board, then we do that, and if we have to call for a meeting, then we do that. Sometimes we just find a meeting room as well."* - Scrum master 1

This quote indicates that despite having a board, sometimes the activity required having a room. The low threshold for getting a meeting room was therefore important for the teams. The meeting rooms therefore also functioned as an important place for horizontal personal mode of coordination.

Task uncertainty

Despite not having found so many trait of task uncertainty within the project, some uncertainty regarding tasks always may exist. Impersonal mode of coordination was previously shown to not be the best mode for handling these task uncertainties, as having plans and schedules for everything is hard when the project gets large and complex. This also contradicts with the agile method of developing software projects. The personal mode, however, could be found to suit the situation of the uncertainties regarding tasks that do appear better. The personal mode was actually found to be valuable for the Case for handling coordination and task uncertainty. As Developer 1 stated:

> *"...you talk to the functional as it appears, I believe it is clever".*

This underlines the ease of just being able to walk to the person who sits on the answer to what one is uncertain about. This highlights the necessity of having highly available employees who can answer questions regarding the tasks. The Case has solved this by having one functional and one technical architect connected to each team and available for these types of questions. The value of being placed near the team has been noticed by the architects itself:

> *"when they see one an other it is much easier for a developer to: I can probably just ask or that they overhear any discussions."* - Other 3

The quote also brings up the repercussions of being able to just walk a short distance to ask for clarification. The talk between two people when being collocated is often overheard by others. This may be good for others who might be uncertain about the same thing, or may be the others haven't started to think about the uncertainty. Further, by hearing the discussion others with different competence fields might also get clarification about the task, or they might even join the discussion giving the uncertainty several perspectives.

Within the Case it was an ambition that the teams should take more ownership of the tasks they were handed, as it was important that the employees felt empowered. This often led to an increased motivation and a better environment:

> *"...one tries to get them to understand more and take more ownership, and to understand more of the total and more of the bigger picture."* - Other 3

The encouragement of the teams to fix their own issues, having as independent teams as possible, led to coordination through mutual adjustment across teams and the team members. Since the Case was able to coordinate even when tasks were uncertain, this can be interpreted to that the use of personal mode of coordination has been somewhat successful. The teams manage to deal with task uncertainty mostly through informal communication and discussion, which resulted in the teams being close to self-coordinated.

The informal communication in a co-located environment opened up the conversation to others, which resulted in several points of views on the issue and knowledge sharing. This personal mode of coordination might therefore be the most important mode for solving task uncertainty. The use of this mode is also expected to increase during the project, as people tend to get to know each other better and therefore knowing who to ask for the different types of issues that may arise.

Task interdependence

In impersonal mode of coordination, was horizontal communication together with planning in front of the start-up meeting found to be important for task interdependence. The planning helped in foreseeing dependencies; schedule the tasks after their dependencies; and arranging the tasks to the different teams. In addition the horizontal communication was considered important to get a better overview of what other teams and team members were working at.

Another factor that was found to be valuable in order to coordinate task interdependence, was the use of informal communication and the teams working ad-hock. The use of informal commu-

nication helped the teams and team members to get a better insight into other teams and team members work through one-to-one communication, but also by over hearing others talking. The scrum masters expressed that it was easy to solve unforeseen dependencies by talking to other scrum masters:

> *"We see dependencies in Jira, or often we just talk about it, both work pretty well. It is easier to talk than to look at Jira... Sometimes I even attend the other teams Stand-ups"* - Scrum master 2

In addition the threshold for attending other teams' stand-ups was also low, due to co-location and stand-ups being held in front of the board for everyone to see.

Because of this, personal mode of coordination seemed to be of importance also when it comes to task interdependence. If there were any unforeseen dependencies, or dependencies that were foreseen that had to be coordinated with some other teams; the scrum master or a team member could easily just talk to the others who the task concerned. Through this informal communication and working ad-hock strategy, these dependencies were easily solved.

Therefore the horizontal personal mode of coordination together with mutual adjustments face-to-face, had an important role of solving dependencies. This decreased the necessity of having a complex hierarchy for decision-making and coordination and therefore making it possible to function more agile. Handling task interdependence can most often take place at a lower level because it often concerns uncertainties or small issues that can be clarified fast with the use of personal mode of coordination at low hierarchical levels.

Size of work unit

Personal mode in terms of unit size concerns the use of personal mode of coordination in context of the size of the project. The interviews illustrate an environment that was struggling with coordination due to the size that the project had grown into. At the time the interviews were held, the teams consisted of 8-10 people and the project consisted of four teams, which was found to be increasingly challenging in terms of coordination and having fully control.

> *"It is clear that it is hard to have fully control over so many people... It is hard for scrum master to be active, it becomes too many to follow up."* - Scrum master 1

> *"...at the biggest we were 10 people at our team, which is quite many, then it is also hard for the scrum masters to participate actively, then you have to let people handle their*

own problems." - Scrum master 2

"The teams have become quite much bigger... It is hard to coordinate commits." - Other 1

From the quotes above it is clear that at least two teams had Scrum masters who found it hard to participate in the degree they were comfortable with. When two teams have come to the same conclusions, one can assume that it also concern others within the project. Both the scrum masters also found the teams to be too big, being difficult to have fully control. In addition the construction manager (Other 1), found it hard to coordinate the commits to the code base with so many people committing to it at all times.

The size of the teams is also challenging in terms of having a good environment within the teams and having good relations to everyone. Where a good environment often leads to greater use of horizontal personal mode of coordination. Also the teams experienced that their teams continued expanding, having to integrate new people to the teams. This lead to further challenges, leading to reduced efficiency.

During the summer the teams got quite big. Because then the new employees who are fresh from school arrived too, so then one always wanted to fill up with new resources, and then they had to be placed somewhere, and then it was not easy to integrate the new once with in the team..." - Scrum master 2

"It affect the efficiency when adding people. It is not that you do not produce, but the others are intended to use time to get the added person integrated." - Other 1

"It got so big that the team members formed small sub groups within the team" - Developer 1

The use of personal mode of coordination is often based on trust and good relations, which grows over time. When new people are added to the teams, or a team gets divided into new teams, one will have to relate to new people, starting over again with building relations and trust. This may impair the use of personal mode.

Further, as the project increased there was also an increased need for knowledge sharing. Knowledge sharing often take place through personal mode of coordination and horizontal informal communication. The Case managed to handle this through rotation of teams depending on which tasks they were working at and their dependencies. It was important for the project that the teams did not grow into their seats, but instead got used to moving around form time to time.

> *"It has been a lot of moving around. We move all the time. So it is what people say that you can't get to comfortable with your place."* - Scrum master 1
>
> *"...one important thing is that we move around people within the teams, so that no one get stuck as a team. Through moving around people one get to know the old ones. Then it is easier to talk together, and connect them within the teams, and then the informal talk proceeds to"* - Other 1
>
> *"So both the solution and the teams have gotten a lot of new people, so they have grown a lot since we started... All of that affects the development. You have to get some time to get to know the others on the team, and such."* Developer 1

The coordination of people tends to get easier as one gets to know the people that shall be coordinated. However, it is easy to be too comfortable with the team one belongs to, and over time this may lead to the teams working in its own "bubble". By shuffling around the teams, adding new team members and splitting up already existing teams, one prevents the "bobble". Instead the new people bring new knowledge and capacity to the teams. In addition, the moving around helps in knowledge sharing and getting to know other people across the teams, building up a better working environment.

The role of personal mode of coordination in terms of unit size can be said to have grown to become of high importance for the project. When the people got to know each other better, less people were added to the project and employees tended to get used to the changing environment of the Case, this increased the use of personal mode of coordination.

The increased use of personal mode of coordination can lead to an increased efficiency, as mechanisms used in impersonal and group mode tends to require larger effort, being more time-consuming. As the project evolves, the team members are expected to be more confident and know the others better, leading to an increase in trust and better relations across team members and teams. This will therefore decrease the need for the other modes of coordination and increase the use for personal mode.

5.2.3 Group mode of coordination

Just as with personal mode, group mode of coordination also takes place through feedback and mutual adjustment. There is however, a difference in the arenas that the feedback and mutual adjustments take place, as group mode takes place through scheduled or unscheduled staff or

committee meetings with several people attending the meeting. This type of coordination requires more planned communication than the two earlier modes, as it often involves change within the organisation (Van de Ven et al. (1976)). Examples of this type of coordination are found in Section 3.2.3.

This mode is, because of its focus on groups, an important mode for large-scale agile software development projects. The team members working at the Case also noticed the importance of the mode.

> *"Both being coordinated between teams and within teams are important...You also have to understand that you are a part of a bigger context too"* - Scrum master 1

Even though having meetings that involved a larger group of the project was important for coordination, the formality of the meetings changed as the project evolved:

> *"Yes, it was many meetings at the beginning, then it got much more informal coordination later."* - Other 1

The quote indicates that it was found necessary to arrange more formal meetings at the beginning of the project. It also indicated the agility of the project, and being able to adjust the formality as the project evolves. However, despite the change of the formality, the number of coordination arenas across all teams and units was not found to have decreased so far.

> *"We have had the same coordination arenas, but the content of the arenas have changed."* - Other 1.

On the other hand, the content of the arenas had changed. This was because they had to adjust the agenda and content after the tasks and the people working with them. Resulting in more efficient meetings and the ability to adjust after the current needs. One of the scrum masters found it important to adjust, as it was possible to adjust to the current need:

> *"Yes, it is very clever to adjust, and in that phase it was test that was in centre."* - Scrum master 1

The use of group mode throughout the project was important for the Case. It was during the coordination arenas that they could collectively inform the teams and team members covering informational needs that were not covered otherwise. And it was also at these arenas that the central unit could get feedback and achieve mutual adjustment. The meetings that were arranged took place at different hierarchical levels, which included team level, Scrum-of-Scrums or Metascrums (Where Metascrums includes the whole project). Having this division of meetings was consid-

ered as beneficial, as if everyone would have attended all the meetings it could result in too many people attending the meetings, resulting in time consuming meetings and too many to relate to. In addition there would be too many meetings to attend to in order to cover all the needs for information sharing and problem solving.

Despite the Case being able to adjust and divide the meetings, some meetings were found to have larger potential for improvement. The planning meeting was one of these, as Developer 1 stated:

> *"I feel that you dont always get enough understanding of the task during the meeting."*

This indicated that the meeting was not used as efficiently as it should. The purpose of the meeting was also to give the team members an understanding of the tasks. There was a need for such meetings, but they didn't work as intended.

Task uncertainty

The arenas found to be important for task uncertainty in terms of group mode were the start-up meeting and planning meeting. It was during these meeting that the tasks were first introduced and further described to the teams and team members. During the Start-up meeting the teams got introduced to the technical and functional characteristics of the tasks, which could also be found at Jira after the meeting and available for the planning meeting to begin:

> *"First we have this joint meeting for everyone, then we go to our teams after wards. Then I have usually cloned the tasks into our environment at Jira."* - Scrum mater 1

During the Planning meeting to each team, the Scrum master was intended to go through the solution description to each task that the team had planned to fulfil during the sprint. This activity was intended to help reducing the task uncertainty.

> *"It sort of depends on the tasks too, but we take a round where we look at the plans, absence and such administrative to look at capacity. Then we have to fill out a form that is pre-made, before we start to look more specific at the solution stories and brake them down, and estimate."* - Scrum master 1

It is during the Start-up meeting that the teams get an overview of all the tasks that shall be finished during the sprint. In addition the teams also get a feeling of who is working with what, which is valuable if it appears that there are any dependencies between teams. When each team

retreats to their seats, team members gets a further understanding of exactly what they shall work on and approximately the scope and difficulty of the tasks.

However, not all the teams found the Start-up meeting as helpfully as it could be. As the quote from Developer 1 states: *"I feel that you dont always get enough understanding of the task during the meeting."* Because of this, the importance of the Planning meeting increased even further, as it was at this arena that the teams could take their time to discuss and clarify tasks that the team members found difficult to understand.

Having these arenas when developing large-scale agile software development is important, as one is certain to meet some task uncertainty. Because of the agile method used, all the tasks shall not have to be planned in detail at the beginning. Meaning that there shall be room for change and deciding the path for the project during the development, and this will lead to some task uncertainty. Handling this in an effective and efficient manner is therefore crucial in order to succeed with coordination and finishing the project successively. Group mode of coordination therefore helps coordinating the task uncertainty, as it is helpful when informing and communicating across teams.

Task interdependence

When considering task interdependence and group mode of coordination, there were also several meetings considered to be important. In addition to the Start-up and Planning meeting that were important for task uncertainty, Scrum-of-Scrum meetings were also found important for task interdependence. It was during the Start-up meeting that all the teams first got introduced to the dependencies between tasks on a board. This meetings was also highlighted as important by the construction manager:

> *"The Start-up meeting is important. It is during that meeting that everyone gets introduced to the dependencies."* - Construction manager

Introducing the dependencies during the Start-up meeting was found important, as they gradually found that there was a need to treat the dependencies more formal:

> *"We have seen the need to treat dependencies more formal than what we did at the beginning"* - Construction manager

The reason for this formal treatment was that they experienced the teams and their team members not reading through the dependencies thoroughly enough.

> *"One shall read through what is written, and what is uploaded, but everyone is not as good at reading through what is written in the solution description."* - Construction manager

Having these arenas for declaring dependencies is of importance for large-scaled agile software development project. It is during the Start-up meeting that the teams can see which other teams they are dependent on for their tasks to be finished. It is also during the Planning-meetings that they can discuss the dependencies even further, finding solutions to how they shall coordinate the dependencies the team has, and when they shall be coordinated. The Scrum-of-Scrums were mostly used to follow up on the dependencies between teams and coordinate when each team needed something from another team:

> *"If there are any specific dependencies between stories, then we get information about how it is going..."* - Construction manager

The meetings used to gather the teams and team members were found necessary for the project to coordinate dependencies. This illustrates the need for group mode of coordination in such types of projects. Giving an overview of all the dependencies, and which teams the different teams are dependent on could be difficult by the use of only personal mode. In addition unexpected dependencies could easily be clarified and coordinated through Scrum-of-Scrums and group mode of coordination.

Size of work unit

The size of work unit can indicate a need to use group mode in order to coordinate efficiently. The size of the Case also increased the need for use of group mode. It was important for the Case to have arenas where one gathered all the teams and team members. It was during these arenas that the project leader could make sure that the information was delivered to everyone, and had the possibility to get feedback and achieve mutual adjustment in plenum.

The use of meetings at different levels has been important for the Case. Through meetings at different levels, the Case achieved more effective and efficient meetings. Only the persons who the meeting concerned or those who could gain from the information sharing, attended the meeting at the different levels. At meetings where only representatives from the different teams attended, the representatives were responsible for communicating the information from the teams and to the other, and from the other teams and central unit and to their own team again.

The communication at the Scrum-of-Scrum meetings that the researcher observed, was experienced to go fluently, with low shoulders and easy communication between project leader, scrum

masters, construction manager and test manager. This observation could also indicate that this was a standard for the Case, indicating a good environment, where the talk went fluently, but still was focused on the subject.

When the project increased in size, it was found to be important to make the tasks as independent as possible. The architects used a time-line in order to avoid too many dependencies taking place at the same time. Where they tried to coordinate for when the different teams should be working with a task that several teams were dependent on. The tasks where first introduced at the Start-up meeting, where all the teams and their team members got to see a map of the dependencies. It was also during this meeting that the teams got an overview of the other teams which they were dependent on during that sprint. The use of this time-line, the dependency map and start-up meeting were therefore important for a successful coordination as the size of the work unit increased.

Even though there was a need to gather all the teams and team members, and have specialised meetings, the use of formal meetings changed throughout the project. The solution architect stated that there are *"fewer formal meetings and more running cooperation across the solution development."* This was indicating that the degree of unscheduled meetings has increased despite being a larger unit than at the beginning of the project. This is interesting, as the need for more formal meetings often follows when the unit size increases.

Chapter 6

Discussion

Through an exploratory case study conducted on a real life case study, with the use of the Van de Ven model, this research has given valuable insight in how large-scale agile software development can be handled in practice.

The research question of this thesis is: *"How can large organisations manage inter-team coordination of large-scale agile software development projects in practice?"*. In this thesis large-scale has been defined as projects or programmes which involves more than one, but less than ten development team. From literature it is known that the scale of such projects leads to challenges, as the complexity tend to increase compared to small-scale agile software development projects. This increased complexity often leads to an increase in task uncertainty, interdependence and work unit size, which are the factors that was found by Van de Ven et al. (1976). The findings also showed some of the characteristics found by Dingsøyr et al. (2017), although they investigated a project of very-large scale. Table 6.1 below summarises the findings found through this thesis. These findings will be discussed in this chapter.

Table 6.1: Main findings in the Case of factors and mechanisms used for improving agility

Area and mode	Findings of factors and mechanisms
The Case	Matrix structure, continuous delivery and a number of coordination arenas
Impersonal mode	Central unit, structure and standardisation of work, specialised teams and co-location
Personal mode	Informal communication
Group mode	Meetings at different levels

The discussion will be structured in terms of Van de Ven's model of coordination and some parallels will be discussed when relevant, where the main focus will be on the findings related to the research question. Before discussing the results with regards to the three different modes, I have explained the factors' relation to the modes and discussed their importance. Lastly, I have evaluated the Van de Vens model itself as well as relevant limitations of this thesis.

6.1 Factors and their relation to the three coordination modes

The factors of task uncertainty, task interdependence and work unit size and how Van de Ven's modes changed depending on these factors were introduced in Section 3.2. The complexity of coordination when projects get large has been recognised by several scientists, Kraut (1995) among others highlighted that certain traits require extensive coordination mechanisms. These traits include the factors also found by Van de Ven, namely: large work unit size, interdependence and uncertainty regarding tasks. Through the study of the Case it has been found several ways of how the Case managed the different factors with the use of the coordination modes:

- *Task uncertainty:* This factor has been managed through several modes, where all of them was found to be used in some degree. However, the personal and group mode of coordination was the mode that was used the most when handling task uncertainty. The impersonal mode was not found to be the best way for handling task uncertainty when the uncertainty was already present. One shall however not forget the importance of the impersonal mode through task description in order to prevent the uncertainty from the beginning.
- *Task interdependence:* All the modes has also been important when handling dependencies between tasks. Where their importance varies throughout the project, depending on which stage they are working at. An interesting aspect that is worth noticing from Chapter 3, is that several sources claim that if there are no dependencies, then there are no need for coordination either (Osifo (2012), Malone (1988)). This indicate that if there are no task interdependence, then there shall be no need for coordination. However, task interdependence is difficult to avoid, when projects gets large, which often leads to a need for several coordination mechanisms and modes for handling them.
- *Work unit size:* Also all the modes have been of importance for the Case when handling work unit size. Where all the modes have played an important role at different stages, the most important modes can be said to be impersonal and group mode of coordination. These modes were important in order to have arenas where information was given to everyone,

but also have plans, schedules and platforms where one could get an overview of the project and find information when it was needed. The mechanisms used to handle unit size, are probably more comprehensive than those one will find in an agile project of small scale.

The findings state that all the modes are always used in some degree when handling all the factors. This is also corresponding to the theory and hypotheses of Van de Ven. However, the degree of the use of the different modes was expected to vary depending on the factors, where the findings found in this study support the hypotheses formulated by Van de Ven, except for Hypothesis 3. This hypothesis states that the use of group mode of coordination is expected to decrease when there is an increase in work unit size. In the Case it is actually found an increased need for group mode as the work unit size increased.

However, the findings highlight the importance of considering the factors and their influence on the different coordination modes. Also the coordination modes are found to be important for handling the factors as they evolve throughout the project.

6.2 Van de Ven's model

6.2.1 Impersonal mode of coordination

The Case was found to use mechanisms from impersonal mode of coordination when handling all the different factors. Some factors were however found to be better handled with the use of other modes. From the results it was found that the Case used impersonal mode through the use of standardised solution descriptions of tasks, developed by the architects. Also because these solution descriptions where developed in advance of the sprint, this indicated that the Case also did a lot of planning ahead. In addition the use of Jira and Confluence were found important for coordination, as well as the physical task board to each team and the use of specialised teams. The Case also used impersonal mode of coordination through the use of its matrix structure and defined roles, having a central unit responsible for the coordination and arranged meetings.

The findings indicate that the impersonal mode of coordination was important for the Case in order to coordinate properly. However, it also reflects a project where planning is a big part of the project. This contradicts with the agile manifesto that states that one shall rather respond to change then following a plan (Beck et al. (2001)). Planning is also something that is not considered by Strode et al. (2012) to be a mechanism of importance when coordinating small-scale projects.

In order to keep the agility it is important for the Case that they have a balance between planning for coordination purposes and still work agile. Finding this balance was also claimed by Dingsøyr et al. (2017) to be important in large-scale agile software development projects. This is, however, also the difficult part with impersonal mode, as one could easily end up removing the agility with too much planning, scheduling and routines. In addition, the thorough planning and solution descriptions can also be seen as problematic, as it might become a bottleneck for the tasks. As the teams have to wait for the solution description to be finished instead of being able to just pick tasks from a queue that is already finished. In the Case, they used more time on the solution description than what a small agile project would have done. When a small project would only include a short description in their solution description (Rising and Janoff (2000)), the Case had a description including several tabs with information. This thorough planning by the Case was also found to lead to small delays. In a large project even small delays can cause significant decrease in efficiency, as it might delay other units in the development process as well. However, with the increased size of the project also the complexity of the project increases (Larman and Vodde (2010), Paasivaara et al. (2012), Bick et al. (2008)). With this increased complexity one will have a need for more thorough solution descriptions, decreasing the decree of task uncertainty. If the solution descriptions of the tasks were less thorough, then this could possibly also have led to an increase in task uncertainty, which could also have caused delays.

It was important for the Case to start with structure, routines, scheduled meetings and plans all the way from the beginning of the project. This was shown through the results from the interviews, where the only meeting arena, of the 16 found, that appeared to have been added towards the first release was the technical review. Compared to small-scale agile projects which only operates with four ceremonials in one sprint (Sommerville (2010)), the number of arenas used in this large-scale agile software development are four times as many. The increased number of coordination arenas is however, claimed by Dietrich et al. (2013) to be needed when the projects get large.

From the analysis it was found that the content of these meeting arenas did change throughout the project. A change in the content of the meetings was also found to take place in the case investigated by Dingsøyr et al. (2017), which also found that some sort of structure was necessary for coordination. Having a strict structure from the beginning can often help in order to get an easier start in order to manage difficulties and issues that come up. With structure the teams and their members knows where to begin without having to depend on others. In addition they also know where to turn to, and there are routines, structure and plans on how to handle troublesome situations. The need for some sort of standardisation was also found by Espinosa (2010) to be beneficial, described through Espinosa's classification of mechanistic coordination.

Since the meetings changed throughout the project, resulting in less formal meetings, one can assume that the importance of having a formal tone and as strict structure decreased as the project evolved. This indicates that the use of impersonal mode of coordination was more important at the beginning of the project, which substantiates Van de Ven's model, suggesting that the use of the different modes evolve throughout the project (Van de Ven et al. (1976)). The use of more impersonal mode of coordination at the beginning, leading to constructing a common base and a common vision for everyone, can also relate to the external coordination defined by Osifo (2012) and the Mintzbergs (1989) standardisation mechanisms for coordination. Mintzberg also suggests that there is a higher need for standardisation of work, output, skills and knowledge at an initial phase of the project, in order for the mechanisms to work as guidance. However, because the Case added new people throughout the project, it was important to continue the use of impersonal mode and the mechanisms for coordination throughout the project. The mechanisms contributed to a safe structure, plans, schedules and routines that all new team members could turn to as they were added to the project. The information and overview of the project could in addition always be found on Jira and Confluence. These technical tools were important for both coordination, and in order to integrate new team members to the Case. The use of impersonal mode of coordination throughout the project was not found to be of the same importance by Dinsøyr et al. (2017). They found a decrease in the use of meetings as the project evolved. However, their research did not mentioned that the project had a continuous increase in size, as the case investigated in this thesis had.

As the number of task dependencies increase, it is claimed by Malone and Crowston (Malone and Crowston (1994)) that the need for more coordination also increases. The need for more coordination arenas was also stated by Dingsøyd et al. (2017) to be helpful when they examined a very-large scaled agile software development project. In one of the observations done by Van de Ven et al. (1976), they found that employees tend to be more tolerant towards more directive and structured leadership. This was also something the Case used. The Case had a matrix-like structure with a central unit that was responsible for coordination. This way of structuring the project deviates from small-scale agile projects and the agile approach to software development, which states that there shall be no project leader (Rising and Janoff (2000)).

The project also operated with other defined roles, such as the main architect. The defined roles was however found to be vulnerable, as they could lead to a single point of failure. However, when Scheerer (2014) studied multi-team systems, he found a need for a different way to handle changes and challenges when coordinating than in single-team projects. Also in previous studies on coordination of large-scale agile software development projects, the projects that were found

to succeed with coordination all included some sort of a central unit responsible for coordination (Bick et al. (2016), Paasivaara et al. (2012), Bick et al. (2008)).

The way one chooses to structure the project can have an huge effect on motivation, environment and the flexibility of the project. In order to adopt to change it is also important for the project to be agile. This takes us back to the importance of finding a balance. The use of co-located and specialised teams was therefore important for the Case. This has also been found by Strode et al. (2012) to be important for small-scaled projects through Strodes theory of the need for structure. As Strode finds it helpful to be physically close to the other team members, having available team members and the ability for other team members to perform the same task as other team members. Also Dingsøyr et al. (2017) found it helpful in their research that the teams were co-located. The co-location in the investigated case lead to having all the teams, supporting units and central unit located and available at all times.

In addition, the use of specialised teams gave the team members some sort of ownership to the tasks, as each team where specialised on their domain. This resulted in that tasks related to a specific domain were handed over to the team with that specialisation. Also when the teams were specialised, the other teams knew where to turn to if they had questions related to a specific domain. Having ownership to tasks is seen as difficult when the projects get large, and if there is no ownership it can lead to a lack of commitment. Making the team members feel that they are committed to the project is important, as the lack of it may result in delaying other teams and also the whole project (Melo et al. (2013)).

One can say that inter-team coordination is affected by size and complexity, which is also what Van de Ven et al. (1976) states. However, as the project increases the tendency of forming sub-groups is also found to increase. This is also a problem that the Case had when the size of the teams increased. This was however solved by the Case through splitting up two large teams and arrange them into three smaller teams. Also with the size and complexity the use of impersonal mode tends to increase in order to coordinate the project, which was also predicted by Van de Ven et al. Because of the increased use of plans, schedules, hierarchies and structure one could easily question the flexibility of the project. Does an increase in size and complexity have to result in a less flexible project? The structures proposed in Section 2.4 suggest both LeSS, Scrum-of-Scrum-of-Scrum (SoSoS) and the hybrid approach. Where the structures suggests how one can scale a agile project.

The structure of the Case is most similar to a hybrid approach than both LeSS and SoSoS. Barlow (2011) suggests that a project uses a hybrid approach if it has characteristics, practices or roles

adopted from two or more methodologies. The Case uses an central unit, which is associated with more traditional approaches and the Waterfall model (Royce (1987)). In addition it uses several agile mechanisms such as, Scrum teams, project backlog, stand-up meetings ect. This approach to software development is also an approach that the teams and their team members seem to be satisfied with. However, for answering the question asked, a project can operate more flexible if it chooses to use a structure more like LeSS or SoSoS. Projects with these types of structures are however little researched, and the structures are yet to be proven to actually work in practice.

6.2.2 Personal mode of coordination

Through the results it was found that personal mode of coordination was also used in order to handle all the three factors presented in Van de Ven's model of coordination. This was also expected, as the model states that personal mode of coordination is expected to increase as the task unit size and the level of uncertainty increases. Both are often related to large-scaled agile projects. Personal mode of coordination relates to the use of informal and formal communication, where the informal communication and co-location in the Case were found to be essential for coordinating the project. However, it also became clear that the size of the project was somewhat troublesome, resulting in Scrum masters and others not having the overview that they normally were comfortable having. In addition the teams experienced that the adding of team members reduced their efficiency.

The Case had the luxury of being able to be co-located at one wing of the building, with the contact persons from the customer being located in the other wing. When the project gets large being co-located tends to be difficult, as few buildings have the space. Being co-located was important for the Case in order to have full advantage of horizontal personal mode of coordination. One of the advantages with co-location was the ability to solve task uncertainties fast, as this was handled by walking over to the person with the answer. Kraut et al. (1995) also saw this advantage in their study, where they found that informal communication could lead to solving uncertainties. Also Van de Ven et al. (1976) noted a relationship between informal communication and uncertainties, where they found that a higher task uncertainty leads to an increased used of mutual adjustment, through horizontal channels and group meetings. The importance of being available and share knowledge is also considered significant for coordination of small-scale projects, where its importance was identified by Strode et al. (2012) through its structure aspect and the availability and substituted part of structure.

Another advantage with co-location was that the team members in the Case was found to ask more questions when the persons with the answers were seated nearby. This indicates that they got their task uncertainties solved faster, leading to being more efficient. Dingsøyr et al. (2017) also saw an advantage of being co-located in terms of coordinating quickly. The use of informal communication while being co-located can lead to other teams and team members overhearing the conversations taking place. This can lead to others taking part in the conversation, resulting in several points of views and knowledge sharing. In addition it could give others, overhearing the conversation, a declaration of the uncertainties as well. Personal mode of coordination is therefore an important mode for declaring task uncertainties.

Coordination achieved through personal mode can also be compared to Espinosa's (2010) theory regarding organic coordination. In this theory Espinosa states that both impersonal and personal communication can be used when coordinating and that coordination is *"achieved through feedback or mutual adjustments, which will say mainly communication and interaction"*. This type of coordination can be compared with the informal communication that took place in the Case and the importance of the use of horizontal communication. Several other large-scale agile software development projects that have been investigated earlier stated that one of the difficulties with coordination was the coordination of interdependence and uncertainties trough horizontal communication (Paasivaara et al. (2012), Bick et al. (2016), Bick et al. (2008)). This may indicate that the lack of horizontal communication is important for successful coordination, as the lack of it could potentially cause unresolved uncertainties and bottleneck for dependencies. This highlights the importance of personal mode of coordination when coordinating.

Small-scale agile projects are encouraged to take decisions at team level, as there shall be no need for a project manager (Rising and Janoff (2000)), and so are large-scale projects if they want to work agile. The use of personal mode might have an even more important role when uncertainties appear due to the teams having to be more independent. This is because often other team members or functional units have the answers. If the project could solve these uncertainties right away, instead of having to call for a formal meeting every time they appear, the project might become even more efficient. Solving these uncertainties calls for the use of informal communication at inter-team level. This was also a method used by the Case. The likeliness of uncertainties appearing across teams regarding tasks and work routines is hard to avoid. This was identified by Kraut et al. (1995), where they found that it is highly likely that something is not predicted in advance in large projects, as people have different opinions regarding how to solve the different tasks. Kraut et al. (1995) also states that if the coordination of uncertainties is poor, then this will lead to project failure. It is therefore crucial for success to manage the coordination of uncertain-

ties, and the use of personal mode is an advantage. This was found by Van de Ven et al. (1976), as the use of personal mode was found to be more efficient than impersonal and group mode of coordination. By declaring uncertainties and dependencies through horizontal communication face-to-face is often more efficient than having to arrange a meeting. The communication is also important at team level; as it helps in spreading the same vision, and for inter-team level; as the declaration of uncertainties and solving dependencies are important for progression and shall be discussed either with the use of impersonal or formal meetings. In addition the interaction that takes place through informal communication may result in a better working environment, as the employees get to know each other better. The improved environment might also result in more motivated team members, leading to an increased efficiency.

The use of personal mode of coordination is expected to increase throughout the project. This is because of one factor that is hard to control in the beginning, namely *trust*. Trust is stated by Lehtimaki (1996) and substantiated by Moore et al. (2008) to be vital for larger project, but is also harder to achieve. This was also shown in the Case, where it was found that there was a decrease in structure and formality, and an increase in informal communication as the project evolved. This can be a result of an increase in trust among the employees and a more relaxed and open environment across teams and team members. It is however, found by Curtis (1988), that the informal communication cannot be expected to work in all large-scale projects. In his research he states that it is actually more likely that it leads to communication bottleneck and breakdowns. This didn't happen in the Case, but it is important to be aware of the risk, as the likelihood increases with the number of people involved.

One challenge that the Case did face was related to the task unit size. The Case was intended to grow into a larger project and used a strategy where they continuously grew. As they grew the Scrum masters expressed that they didn't had fully overview of the project. It became too many team members to have fully control over everyone. The use of personal mode might therefore have been even more important, as dependencies and uncertainties towards tasks increased and could easily be clarified through an informal conversation. Also the team members mentioned that they felt that the efficiency decreased when new team members was added to the team. This is natural, as the team will have to take their time to get to know the new team members and train them. This takes times and results in the persons who are specialised and trained uses more of their time training a new member, leading to reduced efficiency. This method of integrating new team members is however found to be beneficial, as the team members get to know each other fast and the new members might bring new inspiration to the team. The trust that have been built up by the old team members might however be reduced when the teams become so large that it

is time to split the team and make new ones. This might lead to a decrease in the use of personal mode until new trust has been built up, both to new members and to new team members who have come from other teams. I did however fail to find any research on the effect or consequences of adding new members to the project throughout the project. This aspect indicates need for further researched.

Through this analysis it has appeared that personal mode of coordination has been crucial for the Case in order to coordinate properly. Through the results it appeared that the use of personal mode of coordination increased throughout the project, as the team members got to know each other. It was also found that the use of personal mode of coordination helped in increasing the efficiency of the project and assured progress. However, is it possible that too many people in the project will lead to a communication bottleneck and reducing the agility of the project? Even in the Case it was mentioned that the project *"got quite big"*. Is it possible to gain trust to everyone? Will this affect the agility and the coordination of the project? In Van de Ven's model of coordination it is stated that when size increases, the participation of the employees often decrease and more mechanical methods for coordination is used. Trust is also found to be harder to achieve when the projects gets large. The decreased participation, use of mechanical methods and fewer to be trusted can lead to reduced agility, thereby also reduced motivation by the employees and reduced efficiency. Therefore it can be assumed that the size do matter in how agile the project will be, indicating that some adjustments has to be taken.

6.2.3 Group mode of coordination

Even though group mode requires more planned communication as it often involves change (Van de Ven et al. (1976)), it was still used by the Case throughout the project. This mode was therefore also found to be important for managing coordination. Through the results it was stated that the use of group mode through meetings, which gathered all employees working at the project, and meetings at different levels, was an important mechanism for the Case. Van de Ven et al. (1976) also substantiates this, as it is stated in the hypotheses (See hypothesis 1 and 2 in Section 3.2.4) that both an increase in task uncertainty and in interdependence will lead to an increased use of group mode of coordination. Both an increase in task uncertainty and interdependence is related to projects of large-scale, which is different from small-scale projects. Small-scale projects does not experience the increase in interdependence, on the other side, could uncertainties appear also in small-scale projects. Strode et at. (2012) claims that small-scale projects also need some sort of arenas where they gather the whole project through activities in order to gain a common

understanding of the project.

The use of an arena to get a common understanding of the project in the investigated case was also found to be helpful in the investigated case. The arena was used to give everyone an overview of the project, the tasks that one was supposed to finish during the sprint and their dependencies. The investigated case called this meeting the "Start-up meeting", which is also referred to as Meta scrum in this report. It was during this meeting that the teams got introduced to their dependent teams during the sprint. It was also during this meeting that the project leader got the chance to talk to everyone at once, making sure that everyone was introduced to the same information. This was found to be important, as it gathered the whole project. Through the gathering everyone could see that they were part of something bigger then just their team. The introductions of tasks that were given also gave them time to reflect before it was time to go through the tasks and their solution descriptions within the teams. This type of arena was also used in the case investigated by Dingsøyr et al. (2017), together with all the other arenas used they were found to make the coordination efficient. Also Dietrich et al. (2013) found that coordination through group level is efficient through what Dietrich calls "centralised coordination". The importance of gathering the whole project was also recognised by Espinosa (2010) through the theory of Cognitive coordination. Espinosa findings suggests that, when every team member have an overview of the whole project, the people and the tasks, the meetings that are taking place will be more effective.

An important feature with the Start-up meeting was the introduction of the dependencies, illustrating them at a dependency map. Despite using more informal meetings throughout the project, the Case experienced a need to handle dependencies more formal and introduce them properly during the Start-up meeting. This was because they perceived that everyone was not good enough at reading through the solution descriptions by themselves, resulting in more uncertainties. The Case always tried to break down the work into as independent tasks as possible. If there were any dependencies, then they used a time-line for the tasks, so that if possible no team worked at the same task at the same time.

Also if there were tasks who were dependent at other tasks, then the Case seated the teams who had dependent tasks next to each other. These solutions helped in increasing efficiency and reduce the need for coordination, as the teams could easily coordinate themselves as issues aroused. Dependencies are found to be difficult to handle, as the project gets large, and is not found to be a problem in small-scale projects. As the size of the project increase the need to make the task as independent as possible is important as well as clarity, since there shall be no one who are waiting for tasks to be finished. This is supported by Mintzberg's (1989) findings, where he states

that work shall be divided into distinct tasks. He also suggests that dependencies can be solved through mutual adjustment and feedback, which is also relevant for group mode.

The Case also operated with meetings at different levels. In Figure 5.4 shown in Section 5.1 one can see that the Case had meetings at three different levels. This structure was found helpful in order to have arenas for both vertical and horizontal communication. Both vertical and horizontal coordination was also found by Dietrich et al. (Dietrich et al. (2013)) to be important for efficient coordination. Having this structure lead to only people who had an interest in, and had information of value for the meeting were present. Too much time is often used in meetings that is found to be of waste for many. When the project gets large, having these meetings gets even less efficient, as there will be too many attending. The problem of too many attending the meetings was also one of the reasons for the formation of Scrum-of-Scrum and Scrum-of-Scrum-of-Scrum (Schwaber (2007)). It has therefore been found beneficial to divide the meetings, focusing them and make the meetings relevant for those who attend, having only relevant persons present. Often it is beneficial to at least have one member from each team (Paasivaara et al. (2012), Schwaber (2007)). This was also what was used by the Case, where the Scrum masters attended Scrum-of-Scrums, and their technical architect attended the architectural meetings. The people attending can further inform the rest of the team if there is any information that is worth sharing. The information sharing with the rest of the team took often place during the planning-meeting or daily stand-ups. Also Mathieu (2001) found in his research a need for several levels for coordination when coordinating multi-team systems; where he found an effect on work-flow, efficiency, information sharing and learning. During these meetings a fewer number of people attended, which made it a better arena for mutual adjustment and feedback.

However, some meetings held by the Case were found to have potential of being even more effective. The team members did not always felt that they had gotten clarity at the end of the meeting. There was only one person who stated this, and the use of the meetings was not considered as being inefficient by others. However, the use of the meetings could be further investigated through several observations in order to get a better picture of how efficient the meetings were.

Through early recognition of dependencies and clear and planned communication, one can reduce the need for coordination. Even though the Case tried to foresee the dependencies and have focused and relevant meetings they found group mode to be important for managing coordination. The use of group mode helped in order to achieve coordination across teams, but also coordination within the teams. This mode was found to help in declaring uncertainties as well as introducing dependencies. However, even though Van de Ven et al. (1976) states that there is a need for more

formal meetings, as the size of the project increases, the Case only found a need to introduce the dependencies at a more formal arena. The rest of the arenas were not found to change despite an increase in size. The meeting arenas actually became more informal and the Case arranged more unscheduled meetings. The use of group mode can therefore be found to have increased in importance throughout the project. It also seems that the increased size of the project did not affect the formality of the meetings as predicted by Van de Ven (1976). However, the project can be thought to be more formal then what a small project would have been, due to its use of more formal arenas. The use of group mode and personal mode together is, however, valuable as they both achieve coordination through horizontal channels and helps solving problems even faster.

6.3 Evaluation

6.3.1 Van de Ven model

Van de Ven's model of coordination has had a central role in this research. The use of this model could have had an effect on the results; as both observations and interpretations are not independent of theory, and some aspects might have been overseen and therefore not even considered. However, through the model one can get an understanding of how task uncertainty, task interdependence and work unit size can cause variations in the three modes of coordination. Through the analysis of the Case there were also done some observations regarding how these factors could predict the different modes.

One of the observations found, was that the level of impersonal mode of coordination was found to have a higher importance at the beginning of the project. However, the mode was also expected to decrease as the project evolved. That was not found in the Case. The Case actually used the impersonal mode at ha high degree throughout the project. The use of impersonal mode was revealed to be important throughout the project, as the project steadily increased in size. The mode there fore became important for integrating the new project members and for always having some structure to relate to. Even though the level of impersonal mode did not decrease throughout the project, the level of personal mode and group mode tended to increase, as the level of task uncertainty and interdependence increased. The later modes was not only shown to be efficient in handling task uncertainty and interdependence, but also knowledge sharing. Van de Ven's hypothesis 1 was therefore supported, as it was found that when there is an increase in

task uncertainty for organisational units, the use of both personal and group mode seems to increase. The hypothesis also states that the use of impersonal mode is lower when task uncertainty increase.

This was not entirely correct in the Case. The steady use of impersonal mode was found to be correlated with the steadily increase of the project size and not the level of uncertainty. If the size of the project would have been constant from the beginning, the use of impersonal mode could be expected to decrease when the task uncertainty increase.

In terms of hypothesis 2 and how the use of modes adjusts to interdependence, it was found that the level of coordination mechanisms increased when the level of dependencies increased. This was also expected, as the hypothesis states that there is an increase at different degrees in all the modes. In addition the use of agile methodologies helped in coordinating such dependencies, as the organisational structure was more flat, resulting in the teams being able to take decisions and interact more horizontally than if they would have a more hierarchical structure. Even though the use of horizontal communication was found important, the impersonal mode was also of importance when handling task interdependence. For instance, the use of a dependency map being available at Jira and Confluence, and introducing the dependencies formally at the Start-up meeting was found to be of high value for the Case. This indicates that impersonal mode was of importance, but the personal and group mode was found to be even more important when there is an increase in task interdependence. Despite not being able to find a large difference in importance of personal and group mode, these results tend to support the hypothesis 2 of Van de Ven.

The last hypothesis of Van de Ven, hypothesis 3, states that an increase in size of work unit is associated with a decrease in group mode, but an increase and significant increase in personal and impersonal mode of coordination. When considering the findings found when analysing the Case, it was not found that the size of the project affected the use of the different modes in such a large degree as the previous factors. Also the “work unit size” was not mentioned in the same degree as the other factors. However, the size was found to have an impact in some degree, as the use of impersonal mode was found to be of importance throughout the project. This indicates that an increased use of impersonal coordination mechanisms, than if the size of the work unit had been smaller. In addition, it was found necessary to introduce the dependencies at a more formal arena than what was expected from the beginning in the Case.

However, the formality at other arenas actually tended to decrease throughout the project, which was opposite of what was expected. The decrease in formality can be explained by an increase

in trust across team members and teams within the project, as the project evolved. This factor "trust" was not found to be considered by Van de Ven. Further, there was not found to be any decrease in use of group mode of coordination with the increased size. Through the results it was actually found that the level of group mode persisted. From these findings, the hypothesis 3 was only found to be partly supported. These results indicate that the impersonal mode was of higher importance with this large-scale project, and that the personal mode of coordination was of high importance. But there was no findings supporting any decrease in group mode of coordination.

All the factors introduced by Van de Ven et al. were found to be of importance for inter-team coordination in the Case. Also it was clear that different mechanisms were used to manage the different factors. However, there are some drawbacks with this model. It was found to be evident in the Case that the use of the different modes changed throughout the project, depending on which phase they operated in. It was found that the importance of personal mode and group mode increased as the teams and team members developed trust and got an overview of the project. However, as the project steadily increased, there was a need to take time to get to know these new people and for the new people to know the project. This resulted in that the level of personal mode and somewhat also group mode varied. The findings indicates that one task at the beginning of the project might require different mechanisms for coordination later in the project. This is also found by Espinosa (2010). The dynamics of how the use of mechanisms changes throughout a large-scale agile software development project might also be an interesting topic for research.

6.3.2 Research question

The focus of this research is to answer the research question that was introduced in Section 1.3, which is the following: *"How can large organisations manage inter-team coordination of large-scale agile software development projects in practice?"*. Also two sub-questions were formulated in order to answer the main research question. The two sub-questions were as following: 1) *What does the prevailing literature tells us aboutinter-team coordination in large-scale agile software development projects?* 2) *Which coordination mechanisms do the investigated project use in order to coordinate?*

When attempting to answer these questions there were found several challenges, but also some solutions to these, which are worth considering. However, because this research only involves one case, it is not given that these solutions do work for all large-scale agile software development

projects. The researcher believes that they could be of value either way, as it could be used for further research or inspiration.

From the results it was found that informal communication had an important role in coordination, and was found highly necessary when declaring uncertainties and dependencies. Achieving informal communication can however be difficult. Actually Curtis (1988) agrees on this, where he stated that it is difficult achieving efficient informal communication. He also states that most often this type of communication leads to a bottleneck. Its difficulty comes from the fact that it requires building relationships and trust across employees. Doing this when the projects get large is difficult, as the number of people to build a relationship to is large. In order to make it even more difficult, the project often lasts for several years, meaning that the people working with the project exchange. Despite its difficulty, the importance of trust has been found by other scientist as well, to be an important factor for coordination (See Section 3.2.2). It is the trust, that leads the employees to willingly share information and knowledge they contain. The relationship and trust that is built across employees depend on the employees themselves and their ability to develop these.

In order to gain trust and build relationships between employees and then achieve informal communication, one will have to build the right environment. In the Case they strategically developed the teams, and made the teams circulate so that they had to sit next to, and sometimes cooperate with other teams. In addition they had meeting arenas, where different people met both across teams and within a team. Further, when new employees were added, they made sure that the new employees was always placed on a team with people who had been working on the project for a longer time. Lastly, the co-location of the project, was found to be important for the environment and the ease to approach the right team and team members when needed. These findings suggest that the use of group mode of coordination could lead to an increased use of personal mode of coordination.

This leads us to the use of meetings and group mode of coordination within the project. Having meetings both across teams and within teams were found to be important for coordination of the Case. However, in large-project it can often be difficult to know how to conduct them. Meetings also take a lot of time, and sometimes people leave the meeting feeling that it gave them nothing or that it was completely useless. Such use of meetings is highly inefficient, as meetings become a bargain for the employees and could lead to slower progress. So, considering who shall attend the meetings, how often shall they be held and what sort of meetings are necessary, shall be considered in advance. Because of this it could be clever to focus the meetings, meaning that only

competent employees who are relevant for the meeting attend. Also the time of the meeting shall be set in advance. The arrangement of meetings in the Case was solved through the division of levels, where they arranged meetings at different levels. Also if they saw the need for a meeting, the teams and others were free to arrange one. Despite this structure of meetings, some reflected on the planning meeting to have more potential. This was because the team members still had questions after the meeting. This also suggests that the planning-meeting could have been more informal, and be an arena for mutual adjustment, where the employee could have asked all their questions during the meeting. However, as the project evolved, the Case experienced that the meetings got more informal, leading to the use of mutual adjustment and informal communication also during the meetings. This suggests that the use of impersonal mode through structure together with group mode could be a good solution for managing meetings.

The Case used a lot of structure through impersonal mode of coordination. This leads us to another challenge that was found, which was to keep the agility of the project even when it grows larger. When the project gets large, it is challenging to keep the teams as independent as possible, and also be fully coordinated and keep an uniform practice across the project. It is therefore necessary to find a balance between being fully agile and a lot of use of impersonal mode of coordination. Having a balance between these results is a more hybrid approach to software development, which was also the structure that the Case was found to have. The use of too much impersonal mode of coordination could lead to more inefficient teams, slowing down the progress. This is because it often leads to team members feeling less empowered, resulting in reduced motivation. The need for some structure as the project gets large, has been noticed by several scientists (See Section 3.2.1). A solution found by many and also by the Case, is the use of a central unit that is responsible for coordination. It is however, important that this unit does not remove the agility, but also finds a balance between agile mechanism and plans, schedules and structure. Also through a Scrum-of-Scrum set up, the Case made sure that a representative from the team was always present at a meeting, being able to report back to the team.

Not only did the case use a central unit for coordination, but they also used a lot of impersonal mode of coordination, which also persisted throughout the project. In order to keep the agility, they used specialised Scrum teams, who were responsible for their domain. Tasks related to a domain were therefore given to the team with the specialisation required to solve it. The teams therefore got a feeling of ownership towards their domain. Because the teams used Scrum-of-Scrum, they themselves decided which tasks from the given tasks, each team member could work on and operated with other agile mechanisms such as daily stand-up meetings, task-boards, project-backlog ect. In addition the use of more personal mode of coordination increased throughout the

project, leading to task uncertainties and interdependence being declared at more informal arenas. The thorough use of impersonal coordination, led in some cases to delays. This was often related to tasks, and their thorough task-descriptions. The delay suggests that the Case used too much impersonal mode of coordination throughout the project, and that a similar project could take advantage of using a bit less impersonal mode than the Case.

At last the varying use of different mechanisms at different periods throughout the project can be seen as a challenge. Does there exist *a* solution when to use the different mechanisms? Probably not. This is also what makes it challenging. In addition the project needs to be structured such that it is possible to adapt to change. This implies that the project need to work agile. Further, also the team members needs to be willing and prepared for change, as well as acknowledging that at certain periods some mechanisms might work better than others. This was also what took place in the Case. There was seen a need for technical review as the project evolved, which was also accepted and used well by the employees. In addition it was recognised a need to introduce the dependencies more formal, as the project increased in size. These dependencies also changed from only being available at Confluence to also be a part of the content at Jira.

Breaking the results down together with the discussions and considerations above, it becomes possible to determine some main mechanisms that help in managing inter-team coordination in large-scale agile software development projects. One important aspect is that the coordination mechanisms shall be applied in such a way that it does not interfere with the progression of the project, but only in order to make the progress more fluently. The mechanisms recognised to be of highest importance in the Case were:

- **Informal communication:** For information and knowledge sharing
- **Meetings at different levels:** Meetings at different hierarchical levels, leading to time used more efficient, and more focused meetings
- **Central unit:** Use of a central unit responsible for coordination
- **Structure and standardisation of tasks and work:** Gives a safe environment for new project members. I addition it leads to standardisation of work, and assures a safer start of the project, as there are routines, plans and standards for how different things shall be done.
- **Specialised and co-located teams:** Gives teams a feeling of ownership, which results in increased motivation. It also, together with co-location, promotes increased use of personal mode of coordination.

6.4 Limitations

The research has given some interesting findings. There are however, limitations to the research that shall be considered. First of all, this research uses the Van de Ven model of coordination as a frame for the research. Using this model has shown to fit well with the research. Despite this the use of this model also limits the results to be focused on the aspects and factors that the model contains. Thereby, the use of the model may have resulted in that some aspects regarding coordination have been overseen and not included in this research. For a more reliable result one could benefit from testing other models and framing as well, resulting in more precise results. However, if one compare the Van de Ven model to the other coordination models found and mentioned in Chapter 3, the Van de Ven model was the model of best fit for this research.

The use of this model led to the results introduced in Chapter 5. These results reflect the investigated case, but are not necessarily generalisable to a similar project. One finding was that personal mode of coordination was crucial for managing coordination in the investigated case. The mode is highly dependent on the people that the project consists of, and that they are capable of trusting others and build relations. This is also something that has to happen somewhat naturally. One can adjust the environment in a project in some degree, but one cannot force anyone to trust one another or build a good relation. For this taking place naturally might not be the case in all similar projects. A project with a lower use of personal mode could have a different outcome. This type of project could however, have been interesting to research, as it would be interesting to see how it would use the other modes for coordination to compensate.

For this research, fully transcribed interviews were given for analysing. The interviews were semi-structured, where they used three different interview models with a focus on architecture, method adjustments and coordination (See Appendix A). Having semi-structured interviews could lead to more comprehensive and filled out data, but it can also lead to questions being forgotten to be asked. Also it can be difficult for the person analysing to find the information that one expect to find based on the questions. In addition, the selection of interviewees was not random and everyone on the project was not interviewed, but the interviewees were selected based on their roles and availability. This could have affected the results, as it limits the points of view on the project to only a small selection of the project. Further, this selection does not represents all the roles in the project either, which results in some persons meanings are absent from the research. If all the different roles would have been included, this would have given the research extended value and credibility.

Because the interviews where held by another party, and also transcribed by others, the researcher was not biased, and could handle the data neutrally. However, the data has only been analysed by one person, and that may decrease the reliability of the results. In addition, because the researcher was not present during the interviews, the transcribed interviews could have been interpreted wrong and be misunderstood. However, the researcher has tried to handle this by having weekly conversations with one of the interviewers. This gave the researcher the possibility to question the transcribed interviews.

The access to the case has also been limited due to the researcher being located in a different city than the project and because the interviews were held by an external party. The results could have turned out differently if the researcher would have had fully access to the investigated case. If the access was constant, the researcher could have supplied the interviews with more observations and extended interviews.

In advance of analysing the investigated case, a literature study was conducted. The literature on the investigated topic was scare, which could also have an effect on the results; as there are few case studies to compare the analysed case to. The analysis is characterised only by my self, therefore having only one point of view. Also the Van de Ven model was interpreted by only one person, which could also have had an effect on the results. Being more than one person could increase the credibility, as there would be several persons coding the interviews and interpreting the model used. Lastly, I could have overseen important aspects in the transcribed interviews, even though the coding analysis followed a model, which was used thoroughly several times.

Chapter 7

Concluding remarks

7.1 Conclusion

The focus of this study has been on answering the main research question: *"How can large organisations manage inter-team coordination of large-scale agile software development projects in practice?"*. Through this research the following points have been found to be of high importance when a large organisation manages inter-team coordination in a large-scale agile software development project:

- **Informal communication:** In order to handle task uncertainty and manage dependencies, the investigated case draws benefits of the use of personal mode of coordination. The use of personal mode through informal communication increased throughout the project as a result of an increase in trust and relations across employees.
- **Hierarchical meeting structure:** In order for the investigated project to work as efficient as possible, meetings were held at several levels. It was found beneficial that all teams had representatives present at every meeting. Through the representatives, important information could be given to their team members. Further, having a meeting at the beginning of each sprint, with all employees where present, was found to be efficient. The meeting resulted in information sharing and for the employees to get a feeling of being a part of something larger. Lastly, meetings at team-level were found important for gathering a deeper understanding of the tasks, resulting in declaring task uncertainties.
- **Central unit:** When the project became large, it was found a need for more structure in order to coordinate tasks and information across teams. The investigated case solved this

with having a central unit responsible for this coordination.

- **Structure and standardisation of tasks and work:** The use of structure and standardisation was found to be important in the beginning of the project, as it assured a safer start due to its routines, plans and schedules. Also the investigated project steadily increased in size, which introduced the need for having standards and structure for tasks and work throughout the project.
- **Specialised teams and co-location:** Having the whole project located at one floor was found to be of high importance for the investigated project. This promoted the use of informal communication through personal mode of coordination. Also specialised teams were found to increase the use of personal mode, as other employees know whom to ask regarding specific tasks. Specialised teams also gave teams a feeling of ownership, which could result in increased motivation.
- **Maintain the agility and flexibility:** The project evolved and so did the need for the employees. Being able to maintain the agility and flexibility in order to adjust the arenas and methods used throughout the project was therefor found to be important.

7.2 Further research

Through the analysis of the Case I noticed several aspects of coordination in large-scale agile development warranted for further studies. One aspect is how the informal communication in a large-scale project takes place. In this analysis the informal communication was found to be important, but the informal communication has been documented poorly. It could therefore also be interesting to investigate a project where informal communication was not used to the same extent. Maybe also a project with the teams located at different sites too?

This analysis is only based on one round of interviews. The results could have been more reliable if several interviews were arranged. It would have increased the reliability even further if the project was followed from beginning to end. Therefore it would have been interesting to follow a large-scale agile software development project from beginning to end, to see if any coordination arenas or methods changed throughout the project.

I also believe that the SoSoS structures have potentials and should be looked further at, and especially when used in really large-scale software development projects. The field of hybrid approaches was also viewed in this research, but the research found was scarce. Therefore it is of

interest to examine whether it is possible to indicate an optimal combination of traditional and agile methodologies for different types of projects.

Part VI

Appendix

Appendix A

Interview questions

A.1 Architecture

- <Introduction>
- Which tasks did you have in the programme?
- How would you describe the overall architecture in the programme?
- <Timeline exercise focusing on the architecture>
- How do you define software architecture?
- How do you see the relationship between architecture and product characteristics?
- What triggered events in the timeline with respect to architecture? (practices? meetings? decisions?)
- Who were the architects in the programme (static role/dynamic?)
- Who were the main stakeholders?
- How were decisions made? Who made decisions?
- How were trade-offs handled?
- Who gave premises for architectural work? How did you obtain input for architectural work?
- How was the architecture documented?

- Did architectural decisions have an impact on the programme?
- How was architecture communicated?
- How did developers relate to the architecture?
- What main architectural decisions were made?
- Were there changes to the architecture over time?

A.2 Method adjustment

- How do you think you work?
  - Internal within the team
  - Across teams?
  - With technical depth
  - towards other outside the project
- Describe a typical sprint early in the project
- Describe the last sprint
- Have how you work changed throughout the porject?
  - Have the way you work changed over time?
  - What sorts of roles, meetings, description techniques, architecture?
  - Why?
  - When?
  - Who initialise change?
  - It grows versus it is directed?
  - Management directed change?
  - Team?
  - Individual level?
- What is different with the development method in this project?

- Compare this project with earlier projects?
- The agility in the project: some: low degree of agility, other: very agile - what is agile and what is not?

A.3 Inter-team coordination

- Introduction:
  - Background on project and use of information
  - Recording of meeting
  - Anonymity
  - Voluntary participation
  - End of project
- Organization
  - Draw teams and communication arenas between teams
  - Who was sitting where?
- Timeline exercise
  - Important events in the project (in general)
- Retrospective
  - What worked well?
  - What was challenging?
- Inter-team coordination:
  - How was the work organized in your part of the project?
  - What kind of dependencies were there between the teams in your part of the project? (examples?)
  - How were dependencies managed? (examples?)

- What was managed in established fora and what was managed outside of the fora? (examples?)
- Who were involed in managing dependencies between teams? (examples?)
- Did you encounter challenges with managing dependencies? (examples?)
- Did you change the way you managed dependencies during the project? (examples?)
- What practices do you think were most important in order to manage dependencies between teams? (examples?)
- Are there any practices you think had little importance for managing dependencies?
- How did the division of the project into three main parts influence the coordination between teams?
- Were there differences in inter-team coordination across the subprojects?

Bibliography

Barlow, J. B., Giboney, J. S., Keith, M. J., Wilson, D. W., Schuetzler, R. M., Lowry, P. B., Anthony, V., 2011. Overview and guidance on agile development in large organizations. Communications of the Association for Information Systems 29 (1), 25–44.

Batra, D., Xia, W., van der Meer, D., Dutta, K., 01 2010. Balancing agile and structured development approaches to successfully manage large distributed software projects: A case study from the cruise line industry 27, 379–394.

Beck, K., Beedle, M., van Bennekum, A., , Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, J., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., Thoma, D., 2001. Manifesto for agile software development. URL: http://agilemanifesto.org.

Bick, S., Scheerer, A., Spohrer, K., 2016. Inter-team coordination in large agile software development settings: Five ways of practicing agile at scale. In: Proceedings of the Scientific Workshop Proceedings of XP2016. XP '16 Workshops. ACM, New York, NY, USA, pp. 4:1–4:5.

Bick, S., Spohrer, K., Hoda, R., Scheerer, A., Heinzl, A., 2008. Coordination challenges in large-scale software development: A case study of planning misalignment in hybrid settings, 1–21.

Boos, M., Kolbe, M., Kappeler, P., Ellwart, T., 2011. Coordination in Human and Primate Groups. SpringerLink : Bücher. Springer Berlin Heidelberg.

Conboy, K., Fitzgerald, B., 2004. Toward a conceptual framework of agile methods: A study of agility in different disciplines. In: Proceedings of the 2004 ACM Workshop on Interdisciplinary Software Engineering Research. WISER '04. ACM, New York, NY, USA, pp. 37–44.

Crowston, K., Rubleske, J., Howison, J., 2006. A ten-year retrospective. Human-Computer Interaction and Management Information Systems: Foundations, 120.

Curtis, B., Krasner, H., Iscoe, N., Nov. 1988. A field study of the software design process for large systems. Commun. ACM 31 (11), 1268–1287.

Deng, X., Chen, T., Pan, D., 2008. Organizational coordination theory and its application in virtual enterprise. In: Xu, L. D., Tjoa, A. M., Chaudhry, S. S. (Eds.), Research and Practical Issues of Enterprise Information Systems II. Springer US, Boston, MA, pp. 311–316.

Dickinson, T. L., McIntyre, R. M., 1997. A conceptual framework for teamwork measurement. Team performance assessment and measurement, 19–43.

Dietrich, P., Kujala, J., Artto, K., 12 2013. Inter-team coordination patterns and outcomes in multi-team projects 44.

Dikert, K., Paasivaara, M., Lassenius, C., Sep. 2016. Challenges and success factors for large-scale agile transformations. J. Syst. Softw. 119 (C), 87–108.

Dingsøyr, T., Moe, N. B., 2014. Towards Principles of Large-Scale Agile Development. Springer International Publishing, Cham, pp. 1–8.

Dingsøyr, T., Moe, N. B., Fægri, T. E., Seim, E. A., 2017. Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation. Empirical Software Engineering, 1–31.

Dingsøyr, T., Moe, N. B., Seim, E. A., 2018. Coordinating knowledge work in multi-team programs: Findings from a large-scale agile development program. arXiv preprint arXiv:1801.08764.

Dingsøyr, T., Fægri, T. E., Itkonen, J., 2014. What Is Large in Large-Scale? A Taxonomy of Scale for Agile Software Development. Springer International Publishing, pp. 273–276.

Espinosa, J. A., Armour, F., Boh, W. F., 2010. Coordination in enterprise architecting: An interview study. In: System Sciences (HICSS), 2010 43rd Hawaii International Conference on. IEEE, pp. 1–10.

Fredriksen, K. K., 2017. Coordination of teams in large-scale agile software development projects.

Goodman, L., 1961. Snowball sampling. The Annals of Mathematical Statistics 32 (1), 148–170.

Ingvaldsen, J. A., Rolfsen, M., 2012. Autonomous work groups and the challenge of inter-group coordination. Human Relations 65 (7), 861–881.

Kraut, R. E., Streeter, L. A., Mar. 1995. Coordination in software development. Commun. ACM 38 (3), 69–81.

Larman, C., 2004. Agile and Iterative Development: A Manager's Guide. Addison-Wesley.

Larman, C., Vodde, B., 2010. Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum, 1st Edition. Addison-Wesley Professional.

Lehtimaki, H., 1996. Coordination through social networks. University of Tampere.

Malone, T., 01 1988. What is coordination theory?

Malone, T. W., Crowston, K., 1994. The interdisciplinary study of coordination. ACM Computing Surveys vol. 26, pp. 87–119.

Mathieu, J., Marks, M. A., Zaccaro, S. J., 2001. Multi-team systems. International handbook of work and organizational psychology 2 (2).

Melo, C. D. O., Cruzes, D. S., Kon, F., Conradi, R., 2013. Interpretative case studies on agile team productivity and management. Information and Software Technology 55 (2), 412–427.

Mintzberg, H., 1989. Mintzberg on Management: Inside Our Strange World of Organizations. Free Press.

Moore, E., Spens, J., 2008. Scaling agile: Finding your agile tribe. In: Agile, 2008. AGILE'08. Conference. IEEE, pp. 121–124.

Oates, B. J., 2006. Researching information systems and computing. Sage.

Osifo, C., 2012. Organization and coordination.

Paasivaara, M., Lassenius, C., Heikkilä, V. T., 2012. Inter-team coordination in large-scale globally distributed scrum: Do scrum-of-scrums really work? In: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. ESEM '12. ACM, New York, NY, USA, pp. 235–238.

Rising, L., Janoff, N. S., Jul. 2000. The scrum software development process for small teams. IEEE Softw. 17 (4), 26–32.

Royce, W. W., 1987. Managing the development of large software systems: Concepts and techniques. In: Proceedings of the 9th International Conference on Software Engineering. ICSE '87. IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 328–338.

Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering 14 (2), 131.

Scheerer, A., Kude, T., 2014. Exploring coordination in large-scale agile software development: A multiteam systems perspective. Building a better world through information systems : 35th International Conference on Information Systems (ICIS 2014) 17, 69–81.

Schnitter, J., Mackert, O., 2011. Large-scale agile software developemt at sap ag. Evaluation of Novel Approaches to Software Engineering vol. 230, pp. 209–220.

Schwaber, K., 2007. The Enterprise and Scrum, 1st Edition. Microsoft Press, Redmond, WA, USA.

Smith, L., Schwegler, U., 2010. The role of trust in international crisis areas: A comparison of german and us-american ngo partnership strategies. Organisational Trust: A Cultural Perspective, 281–310.

Sommerville, I., 2010. Software Engineering, 9th Edition. Addison-Wesley Longman Publishing Co., USA.

Strode, D. E., Huff, S. L., Hope, B., Link, S., Jun. 2012. Coordination in co-located agile software development projects. J. Syst. Softw. 85 (6), 1222–1238.

Thompson, J. D., 1967. Organizations in action: Social science bases of administrative theory. Transaction publishers.

Van de Ven, A. H., Delbecq, A. L., Koenig Jr, R., 1976. Determinants of coordination modes within organizations. American sociological review, 322–338.

VersionOne, 2017. Version one's 11th annual state of agile report. URL: https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2.

Zmud, R. W., 1980. Management of large software development efforts. MIS quarterly, 45–55.